COMPUTER LANGUAGE™

VOLUME 4, NUMBER 11

NOVEMBER 1987

$3.50 Canada $4.50

PATTERN MATCHING: THEORY VS. PRACTICE

MULTIPLE WORD SEARCHES WITH LFIND

MEGASORT: THE ULTIMATE SORT?

A FORMULA FOR SYMMETRICAL DESIGN

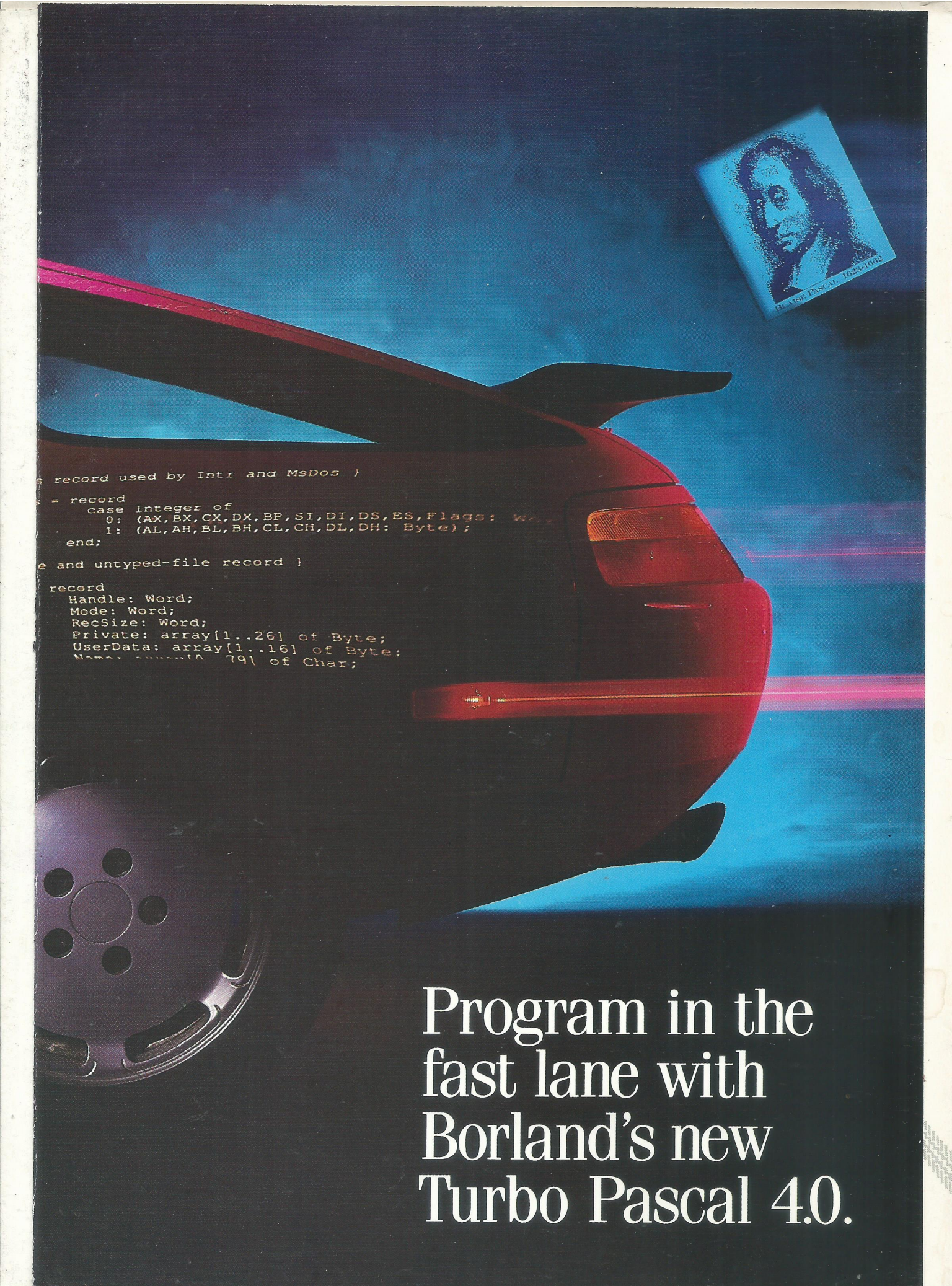
SYNCHRONIZATION IN HARD-TO-EASY DESIGN

ALGORITHMS

M BARRY
9244 TAVERN
WILTON

11

0 74470 17345 8


record used by Intr and MsDos }
= record
case Integer of
0: (AX,BX,CX,DX,BP,SI,DI,DS,ES,Flags:
1: (AL,AH,BL,BH,CL,CH,DL,DH: Byte);
end;
e and untyped-file record }
record
Handle: Word;
Mode: Word;
RecSize: Word;
Private: array[1..26] of Byte;
UserData: array[1..16] of Byte;
of Char;
BLAISE PASCAL 1623-1662
Program in the fast lane with Borland's new Turbo Pascal 4.0.

The fast lane is *fas*

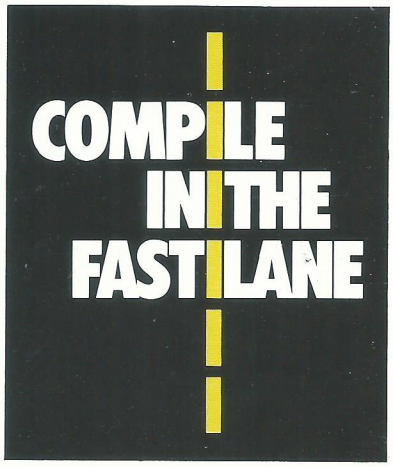
Our new Turbo Pascal® 4.0 is so fast, it's almost reckless. How fast? Better than 27,000 lines of code per minute. That's much faster than 3.0 or any other Pascal compiler and the reason why you need 4.0 today.

Pascal. The fastest and the best.

If you're just now learning a computer language, learn Pascal. If you're already programming in Pascal, you're programming with a winner because Pascal is the worldwide language of choice. Pascal is the most popular language in university computer science classes and with computer enthusiasts who appreciate Pascal's modern programming structure. It's powerful, coherent, easy to learn and use—and with Turbo Pascal 4.0—faster than ever before.

Turbo Pascal: Technical excellence

Commitment to technical excellence and



superiority also means commitment to detail, however painstaking, and that takes time. 4.0's predecessor, Turbo Pascal 3.0 is the worldwide standard, and with Turbo Pascal 4.0, we've bettered that standard. 4.0 is clearly the world's fastest development tool for the IBM® PS/2 series, PC's and compatibles—and the world's favorite Pascal compiler.

4.0 breaks the code barrier

No more swapping code in and out to beat the 64K code barrier. Designed for large programs, Turbo Pascal 4.0 lets you use every byte of memory in your computer. You paid for all that memory, now you can use it freely.

For the IBM PS/2 and the IBM and Compaq families of personal computers and all 100% compatibles.

YES!

I want to upgrade to Turbo Pascal 4.0 and the 4.0 Toolboxes

Registered owners have been notified by mail. If you are a registered Turbo Pascal user and have not been notified of Version 4.0 by mail, please call us at *(800) 543-7543*. To upgrade if you have not registered your product, just send the original registration form from your manual and payment with this completed coupon to:

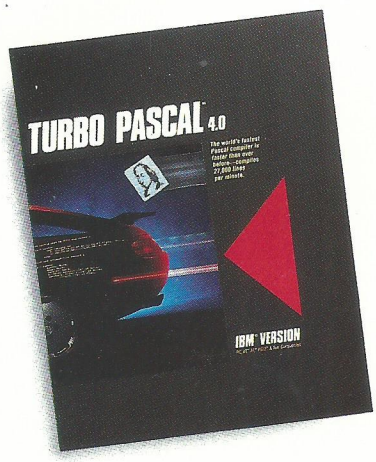
Pascal 4.0 Upgrade Dept.
Borland International
4585 Scotts Valley Drive
Scotts Valley, CA 95066

Name _____
Ship Address _____
City _____ State _____
Zip _____ Telephone ( ) _____

This offer is limited to one upgrade per valid registered product. It is good until November 30, 1987. Not good with any other offer from Borland. Please allow 4 to 6 weeks for delivery of Toolboxes.

Outside U.S. make payments by bank draft payable in U.S. dollars drawn on a U.S. bank. CODs and purchase orders will not be accepted by Borland.

ter than ever before!



4.0 uses logical units for separate compilation

Pascal 4.0 lets you break up the code gang into "units," or "chunks." These logical modules can be worked with swiftly and separately—so that an error in one module is seeable and fixable, and you're not sent through all your code to find one error. Compiling and linking these separate units happens in a flash because your compiling horsepower is better than 27,000 lines a minute.* And 4.0 also includes an automatic project Make.

4.0's cursor automatically lands on any trouble spot

4.0's interactive error detection and location means that the cursor automatically lands where the error is. While you're compiling or running a program, you get an error message at the top of your screen *and* the cursor flags the error's location for you.

4.0 gives you an integrated programming environment

4.0's integrated environment includes pull-down menus and a built-in editor. Your program output is automatically saved and shown in the output window. You can Scroll, Pan, or Page through all your output and know where everything is all the time. Given 4.0's integration, you can edit, compile, find and correct errors—all from inside the integrated development environment.

You'll never lose your mind, because 4.0 never loses your place

Whenever you re-load 4.0, it remembers what you and it were doing before you left. It puts you right back in the editor with the same file and in the same place as you were working last.

*Run on an 8 MHz IBM AT.

**If within 60 days of purchase this product does not perform in accordance with our claims, call our customer service department, and we will arrange a refund.

All Borland products are trademarks or registered trademarks of Borland International, Inc. Other brand and product names are trademarks or registered trademarks of their respective holders.
Copyright © 1987 Borland International, Inc. BI 1159

Please check box(es)	*Sugg. Retail*	*Upgrade Price†*	Serial No.
☐ Turbo Pascal 4.0 Compiler	*$ 99.95*	*$ 39.95*	_____
☐ Turbo Pascal 4.0 Developer's Library	*395.00*	*150.00*	
(Includes Turbo Pascal Tutor and all Toolboxes; must be ordered with Compiler)			
☐ Turbo Pascal Tutor	*69.95*	*19.95*	_____
☐ Turbo Pascal Database Toolbox	*99.95*	*29.95*	_____
☐ Turbo Pascal Graphix Toolbox	*99.95*	*29.95*	_____
☐ Turbo Pascal Editor Toolbox	*99.95*	*29.95*	_____
☐ Turbo Pascal Numerical Methods Toolbox	*99.95*	*29.95*	_____
☐ Turbo Pascal Gameworks	*99.95*	*29.95*	_____
Total product amount		*$* _____	
CA and MA residents add sales tax		*$* _____	
Shipping and handling*		*$* _____	
Total amount enclosed		*$* _____	

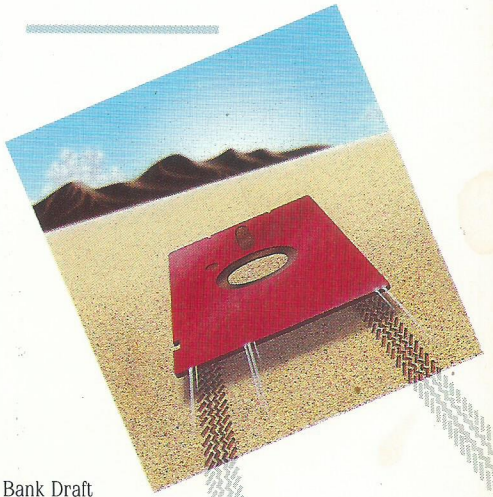
Please specify diskette size: ☐ 5¼" ☐ 3½" Payment: ☐ VISA ☐ MC ☐ Check ☐ Bank Draft

Credit card expiration date: _____/_____

Card # |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|

*In US please add $5 shipping for each product ordered or $15 for the Compiler and Developer's Library. Outside US please add $10 shipping and handling for each product ordered or $25 for the Compiler and Developer's Library.

†To qualify for the upgrade price you must give the serial number of the equivalent product you are upgrading.



CL 11/87

Now's the time for a *fast* decision: Upgrade now to 4.0!

Compatibility with Turbo Pascal 3.0

We've created 4.0 to be highly compatible with version 3.0 and included a conversion program and compatibility units to help you convert all your 3.0 programs to 4.0.

Highlights of Borland's new Turbo Pascal 4.0

- Compiles 27,000 lines per minute
- Supports >64K programs
- Uses units for separate compilation
- Integrated development environment
- Interactive error detection/location
- Includes a command line version of the compiler

4.0 also

- Saves output screen in a window
- Supports 25, 43 and 50 lines per screen
- Generates MAP files for debugging
- Has graph units including CGA, EGA, VGA, MCGA, 3270 PC, AT & T 6300 & Hercules support
- Supports extended data types (including word, long integers)
- Does smart linking
- Comes with a free revised MicroCalc spreadsheet source code

4.0 is all yours for only $99.95

Sieve (25 iterations)

	Turbo Pascal 4.0	***Turbo Pascal 3.0***
Size of Executable File	2224 bytes	11682 bytes
Execution speed	9.3 seconds	9.7 seconds

Sieve of Eratosthenes, run on an 8MHz IBM AT

Since the source file above is too small to indicate a difference in compilation speed we compiled our GOMOKU program from Turbo Gameworks to give you a true sense of how much faster 4.0 really is!

Compilation of GO.PAS (1006 lines)

	Turbo Pascal 4.0	***Turbo Pascal 3.0***
Compilation speed	2.2 seconds	3.6 seconds
Lines per minute	27,436	16,750

GO.PAS compiled on an 8 MHz IBM AT

60-Day Money-Back Guarantee**

BORLAND INTERNATIONAL

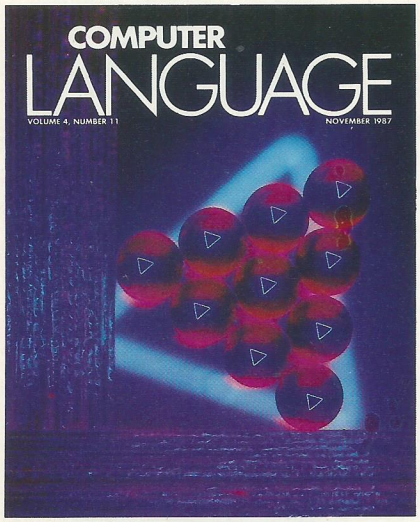
For the dealer nearest you or to order call

(800) 543-7543.

CIRCLE 1 ON READER SERVICE CARD

BORLAND

COMPUTER LANGUAGE™



ARTICLES

Pattern Matching Alternatives: Theory vs. Practice ____ **34**
by Jack Purdum
This article complements last year's Boyer-Moore string searching article, comparing the Boyer-Moore algorithm with Knuth-Morris-Pratt and brute force approaches to large character sequence searches. Purdum points out that theoretical testing of an algorithm's speed isn't always a fair judge of how it will perform in practice, and uses the results of 10 timing tests to illustrate his point.

Multiple Word Searches With LFIND ____ **47**
by E.J. Smith
LFIND offers many advantages over the DOS *FIND* command: it does multiple word searches, isn't case sensitive, displays the line in which a word is found, and shows the line number of the text file. One of LFIND's most attractive features, though, is that it takes the same amount of time to search for 50 words as for one word.

A Recipe for Self-Decorating Eggs ____ **55**
by Clifford A. Pickover
The simple formula presented in this article uses real numbers instead of imaginary ones to create an infinite number of "chaos eggs"—egg-shaped tiling patterns that bear a striking resemblance to the artwork of some ancient cultures.

Megasort: A Distribution Sort ____ **63**
by Steve Heller
Using Turbo Pascal and something called "distribution counting" (described in volume 3 of *The Art of Computer Programming*), the author created a short, fast distribution sort known as Megasort. One of Megasort's biggest advantages is that its execution time is linear in the number of items to be sorted.

DEPARTMENTS

Feedback ____ **7**

Programming on Purpose ____ **15**
Synchronization in hard-to-easy design

Designing With Data Bases ____ **25**
Screen and report generators

Public Domain Software Review ____ **73**
Pascal programming aids and PC-FASTYPE

Exotic Language of the Month Club ____ **79**
Beyond text editors with VAXTPU

Product BINGO ____ **90**

Software Reviews ____ **93**
- Unipress Software's C-macs
- The Software Factory's OPAL
- ESP Software's Command Plus
- Gotoless Software's Bas_C

Classified Connection ____ **118**

Advertiser Index ____ **120**

If you thought all file managers were alike, think again. Think Btrieve.

Btrieve stands out as the one file management system for programmers who are serious about productivity. With Btrieve, your programs can use simple subroutine calls to store, retrieve, and update records. It's the file handling solution rich in functionality to meet your needs now and in the future, as your requirements grow and change.

Bfast. With Btrieve you develop fast applications, with ease. Written in assembly language for IBM PC's, Btrieve uses b-tree algorithms with automatic balancing for fast, efficient file indexing.

Bsafe. Btrieve is the only file manager with automatic file recovery. Two levels of database integrity provide complete fault tolerance in the event of accidents or power failures.

Bflexible. Develop your PC applications with the capabilities you need most. Capabilities such as: unlimited open files, unlimited records per file, 24 indexes per file and a maximum file size of 4 gigabytes. All this with Btrieve plus access from any programming language.

Bconnected. When your applications require networking, think Btrieve. A single version runs on all DOS 3 LAN's, including IBM PC Network and Novell Advanced NetWare. Btrieve is also available for Xenix and other multi-tasking operating systems.

Broyalty Free. No royalty payments on your applications. Ever.

Bsure. Btrieve comes with comprehensive documentation and unrivaled technical support for trouble-free software development. And you can add Xtrieve to supply your end users with even more, a menu-driven report writer and query system.

Think Btrieve. Make your programming performance stand out. Contact SoftCraft.

SoftCraft
A NOVELL COMPANY
P.O. Box 9802, #917
Austin, Texas 78766
(512) 346-8380 Telex 358 200

Btrieve, $245; multi-user Btrieve, $595; Xtrieve, $245; multi-user Xtrieve, $595 (for report generation, add $145 for single-user and $345 for multi-user). Requires PC-DOS or MS-DOS 2.X, 3.X or Xenix. Btrieve and Xtrieve are registered trademarks of SoftCraft Inc.

CIRCLE 2 ON READER SERVICE CARD

COMPUTER LANGUAGE

EDITOR
Regina Starr Ridley
MANAGING EDITOR/ ART DIRECTION
Kathy Kincade
PRODUCT REVIEW EDITOR
J.D. Hildebrand
EDITORIAL ASSISTANTS
Joanna M. Poole, Johanna Kleppe
TECHNICAL EDITOR
John Halamka
CONSULTING EDITOR
Craig LaGrow
CONTRIBUTING EDITORS
Tim Parker, P.J. Plauger, Matthew Rapaport, Leonard Struttman
NATIONAL ADVERTISING DIRECTOR
Ted Bahr
REGIONAL SALES MANAGERS
Mary Christine Gordon
(Midwest—216-493-8733)
Walter Andrzejewski
(East—617-232-5470)
Jeff Eilertsen
(West—415-397-1881)
ADVERTISING SALES ASSISTANTS
Susan McDonald, Linda Pizzotti, Gina Dalpra
SALES/MARKETING ASSISTANT
Carol A. Frederico
DIRECTOR OF PRODUCTION
Andrew A. Mickus
OPERATIONS CONSULTANT
Beatrice C. Blatteis
DIRECTOR OF CIRCULATION
Wini D. Ragus
CIRCULATION MARKETING MANAGER
Jerry M. Okabe
ADVERTISING TRAFFIC COORDINATOR
Kelly B. Keller
WHOLESALE COORDINATOR
Nicola Sullivan
ART PRODUCTION
Anabella Gonzalez

PUBLISHER
Donald Pazour
CHAIRMAN OF THE BOARD
Lord David R. Stevens
PRESIDENT
Marshall W. Freeman
VICE PRESIDENTS
Charles C. Baake
Leigh M. Freeman
Leonard E. Haas
Barbara M. Hampson
Thomas L. Kemp
Wini D. Ragus
CORPORATE FINANCIAL MANAGER
Charles H. Benz
SECRETARY/TREASURER
Miller Freeman III
ASSISTANT TREASURER
Douglas M. Denny

FOUNDING PUBLISHER
Carl Landau

MILLER FREEMAN publications

Editor's Notes

"The study of algorithms is the study of the very heart of computing," wrote Jon Bentley in "An Introduction to Algorithm Design" (*IEEE*, Feb. 1979, pp. 66-78). Algorithms have always been the focal point of *COMPUTER LANGUAGE* editorial; this issue is our third annual issue on the subject.

Donald Knuth's basic definition of an algorithm is "a finite set of rules which gives a sequence of operations for solving a specific type of problem" (*The Art of Computer Programming*, vol. 2, "Fundamental Algorithms," Addison-Wesley). To that definition he adds five important features:

"Finiteness. An algorithm must always terminate after a finite number of steps.

Definiteness. Each step of an algorithm must be precisely defined.

Input. An algorithm has zero or more inputs, that is, quantities that are given to it initially before the algorithm begins.

Output. An algorithm has one or more outputs, that is, quantities which have a specified relation to the inputs.

Effectiveness. An algorithm is also generally expected to be effective...all of the operations to be performed...must be sufficiently basic that they can in principle be done exactly and in a finite length of time by a man using a pencil and paper."

The bottom line, notes Knuth, is that a programmer wants good algorithms in a loosely defined aesthetic sense, judged by such criteria as length of time to execute, adaptability, and simplicity and elegance.

Knuth's insights from *The Art of Computer Programming* are given here to provide a bit of a framework for examining and trying out the algorithms presented in this issue.

"Pattern-Matching Alternatives: Theory vs. Practice," written by Jack Purdum, picks up from where last year's "Searching for Strings with Boyer-Moore" left off. Purdum compares Boyer-Moore, Knuth-Morris-Pratt, and brute force approaches for searching a large sequence of characters for a match on a specific character pattern and comes up with some very interesting conclusions.

LFIND, a list-find program similar to the DOS *FIND* command, is described by E.J. Smith in "Multiple Word Searches with LFIND." LFIND is a fast, multiword search program that accepts a list of words and searches for them in a single pass.

Of course, we couldn't have an algorithms issue without a sorting article. Steve Heller presents Megasort, a distribution sort he claims is no more than a constant factor slower than any other sort and faster than almost any other on large arrays.

Finally, we just had to give you an unusual and slightly off-beat algorithm—one for generating intricate tiling patterns. The algorithm, developed by Clifford Pickover, yields egg-like shapes called "chaos eggs." These chaos eggs exhibit striking similarity to some of the patterns seen in the artwork of ancient cultures.

The best thing to do with these algorithms is jump right in and play around with them. Check their effectiveness. As Knuth says, "An algorithm must be seen to be believed, and the best way to learn what an algorithm is all about is to try it." Have some fun!

Regina Starr Ridley

Regina Starr Ridley
Editor

COMPUTER LANGUAGE code can be obtained from CompuServe (type "GO CLM" at the prompt and you'll be in our SIG), USENET ("mod.sources"), and Bulletin Board Services in the U.S. and Canada. Bulletin board parameters are 8 data bits, no parity, 1 stop bit, full duplex, and 300 or 1200 baud.

Complete codes listings for all issues since the Premier in 1984 are available from the Chicago BBS and CompuServe. Code on the other BBSs goes back six to 12 months.

COMPUTER LANGUAGE code is also available on disk from the IBM PC User Group in Austria. For further information contact them at Hasibederstr. 2/64/6, B.O. Box 40, A-1225, Vienna, Austria. Telex: 1-11010 tzxt, attention: IBM PC User Group.

Chicago, Ill.	(309) 342-5302	Edmonton, Canada	(403) 454-6093
East Lansing, Mich.	(517) 355-3276	Ontario, Canada	(613) 830-1590
Cleveland, Ohio	(216) 486-0348	New York, N.Y.	(212) 980-0770
Worthington, Ohio	(614) 885-9829	Oklahoma City, Okla.	(405) 728-2463
Seattle, Wash.	(206) 848-9232	Long Beach, Calif.	(213) 498-6425
San Francisco, Calif.	(415) 487-0310	Linton, Ga.	(404) 444-6713
Austin, Texas	(512) 346-6671		

COMPUTER LANGUAGE *(ISSN 0749-2839) is published monthly by Miller Freeman Publications, 500 Howard St., San Francisco, CA 94105, (415) 397-1881. Please direct advertising and editorial inquiries to this address. Subscription rates for the U.S. are $29.95 for 12 issues, $39.95 for 24 issues, and $49.95 for 36 issues. Canadian/Mexican orders must be accompanied by payment in U.S. funds with additional postage of $6 per year. All other foreign must be prepaid in U.S. funds with additional postage at $15 per year for surface mail and $40 per year for air mail. All subscription orders, inquiries, and address changes should be sent to P.O. Box 11333, Des Moines, IA 50340-1333. For quickest service, telephone toll-free 1-800-341-SERV (in Iowa 1-800-233-IOWA). Second class postage is paid at San Francisco, Calif., and at additional mailing offices. Please allow six weeks for change of address to take effect.* COMPUTER LANGUAGE *is a registered trademark owned by the magazine's parent company, Miller Freeman Publications. All material published in* COMPUTER LANGUAGE *is copyrighted © 1987 by Miller Freeman Publications Inc. All rights reserved. Reproduction of material appearing in* COMPUTER LANGUAGE *is forbidden without written permission.*

Blaise puts the Accent on C with C TOOLS PLUS/5.0™

Now, just $129 and supports Microsoft C 5.0 and QuickC

Enhance your Microsoft C programming environment with C TOOLS PLUS/5.0™— a new, quintessential library of C functions. C TOOLS PLUS/5.0 from Blaise Computing Inc. puts a prime accent on quickly building professional applications using the full power of Microsoft C Version 5.0 and QuickC. Now you can concentrate on program creativity by having full control over DOS, menus, interrupt service routines, memory resident programs, printer and keyboard control, and more!

C TOOLS PLUS/5.0 prebuilt libraries are ready to use with either QuickC or the Microsoft C Version 5.0 command line environment. Complete documented source code is included so that you can study and adapt it to your specific needs. Blaise Computing's attention to detail, like the use of full function prototyping, cleanly organized header files, and a comprehensive, fully-indexed manual, makes C TOOLS PLUS/5.0 the choice for experienced developers as well as newcomers to C.

Continuous refinement of Blaise Computing's library products has produced a collection of tools that are unsurpassed for reliability, functionality and ease of use. Built upon the widely acclaimed C TOOLS PLUS, C TOOLS PLUS/5.0 includes such highly-developed features as:

◆**WINDOWS**
— Stackable, removable.
— Optional borders, cursor memory.
— Accept user input, formatted output.
— "printf" window-oriented output. **NEW!**

◆**INTERRUPT SERVICE ROUTINES**
— Capture DOS critical errors and keystrokes.
— Install hardware interrupt handlers.

◆**RESIDENT SOFTWARE SUPPORT**
— Install, detect and remove memory resident programs.

◆**MENUS**
— Horizontal and pulldown. **NEW!**
— Lotus-style support. **NEW!**

◆**INTERVENTION CODE**
— Schedule C functions at specified times, intervals or with a "hot key." **NEW!**
— Take full advantage of DOS, even from memory resident programs. **NEW!**

◆**FAST DIRECT VIDEO ACCESS**
— All monitors, even EGA 43-line mode.

◆**PRINTER CONTROL**
— Access BIOS print functions. **NEW!**
— Control the DOS PRINT utility. **NEW!**

◆**UTILITIES AND MACROS**
— Take advantage of DOS file structure.
— Manipulate data types, far & near pointers. **NEW!**
— Access any memory areas with fast "peek" and "poke" macros. **NEW!**

C TOOLS PLUS/5.0 supports the Microsoft C Version 5.0 and QuickC compilers, requires DOS 2.00 or later and is just **$129.00.**

C ASYNCH MANAGER™ Version 2.0 IMPROVED!

C ASYNCH MANAGER is a library of functions designed to help you incorporate asynchronous communication capabilities into your application programs. Version 2.0 has been rewritten especially for Microsoft C Version 5.0 and Borland's Turbo C. Simultaneous buffered input and output to both COM ports at speeds up to 9600 baud, XON/XOFF protocol, modem control and XMODEM file transfer are among the many features supported and is priced at just **$175.00.**

Blaise computing Inc. has a full line of support products for both Pascal and C. Call today for your free information packet.

THE BLAISE MENU

VIEW MANAGER *$275.00*
General screen control; paint screens; block mode data entry or field-by-field control with instant screen access. For C or MS-Pascal.

Turbo C TOOLS *$129.00*
Windows; ISRs; intervention code; screen handling and EGA 43-line text mode support; direct screen access; DOS file handling and more. For Turbo C.

Turbo POWER SCREEN
COMING SOON! General screen management; paint screens; block mode data entry or field-by-field control with instant screen access. For Turbo Pascal.

Turbo POWER TOOLS PLUS *$99.95*
Screen and window management including EGA support; DOS memory control; ISRs; scheduled intervention code; and much more. For Turbo Pascal.

Turbo ASYNCH PLUS *$99.95*
Interrupt driven support for the COM ports. I/O buffers up to 64K; XON/XOFF; up to 9600 baud; modem and XMODEM control. For Turbo Pascal.

PASCAL TOOLS/TOOLS 2 *$175.00*
Expanded string and screen handling; graphics routines; memory management; general program control; DOS file support and more. For MS-Pascal.

ASYNCH MANAGER *$175.00*
Full featured interrupt driven support for the COM ports. I/O buffers up to 64K; XON/XOFF; up to 9600 baud; modem control and XMODEM. For MS-Pascal.

KeyPlayer *$49.95*
"Super-batch" program. Create batch files which can invoke programs and provide input to them; run any program unattended; create demonstration programs; analyze keyboard usage.

EXEC *$95.00*
NEW VERSION! Program chaining executive. Chain one program from another in different languages; specify common data areas; less than 2K of overhead.

RUNOFF *$49.95*
Text formatter for all programmers; flexible printer control; user-defined variables; index generation; general macro facility. Crafted in Turbo Pascal.

LIGHT TOOLS *$99.95*
Windows; ISRs; EGA 43-line text mode; direct screen access; DOS file handling and more. For the Datalight C compiler.

TO ORDER CALL TOLL FREE
800-333-8087
TELEX NUMBER-338139

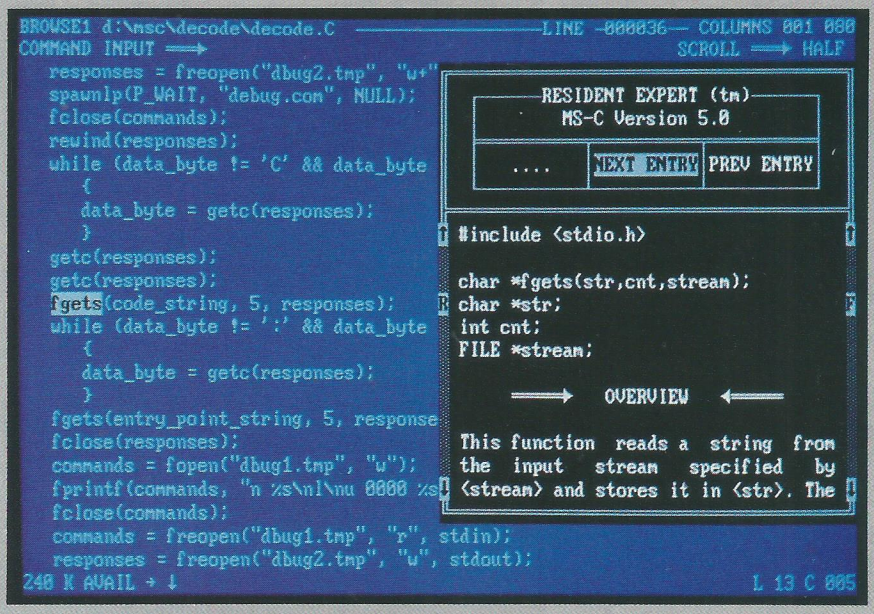
Yes! Send me the prime accent!
Enclosed is $_____ for _____ copies of _____.
☐ Please send me more information on your products.
CA residents add Sales Tax. Domestic orders add $4.00 for UPS shipping, $10.00 for Federal Express standard air.
Name: _____ Phone: (_____) _____-_____
Address: _____ State: _____ Zip: _____
City: _____ Exp. Date: _____
VISA or MC#: _____

Microsoft is a registered trademark of Microsoft Corporation. QuickC is a trademark of Microsoft Corporation. Turbo C is a registered trademark of Borland International.

BLAISE COMPUTING INC.
2560 Ninth Street, Suite 316 Berkeley, CA 94710 (415) 540-5441

CIRCLE 3 ON READER SERVICE CARD

RESIDENT EXPERT Pop-up Reference Guides...



Try One And Get Our MS-DOS/PC-DOS Guide Absolutely FREE!

THE POP-UP REFERENCE REVOLUTION BEGINS

How much development time could you save if you never had to open another PC language or technical reference manual again? What if you could just point at a compiler keyword, assembly instruction, or function name *on your screen* and with a keystroke have complete, authoritative information about language syntax, operands, parameters, examples, and much more?

INTRODUCING THE RESIDENT EXPERT SYSTEM

A growing library of comprehensive, disk resident reference guides about the PC *and* your favorite PC languages. All available instantly through our unique memory resident pop-up access system.

VIRTUALLY EVERYTHING YOU NEED TO KNOW

Each of our *Compiler Reference Guides* contains virtually everything you need to know to program with your *preferred implementation* of your favorite language. Language syntax, all library functions, compiler directives, and error codes are thoroughly documented.

Our *PC Programmer's Reference Guide* documents every PC (and AT) processor instruction and every BIOS and DOS service interrupt. You'll also find tables of keyboard codes, line drawing, ASCII, and IBM character sets, and much more.

THE SPECIALIST'S LIBRARY

Your compiler is unique. That's why our reference guides are *specialized*...each one designed for a particular vendor's language implementation.

Free With Any Purchase!

Our Companion Pop-up Guide To MS-DOS 3.2/PC-DOS 3.3

Limited Time Offer

QUICK DRAW ACCESS SYSTEM

Point-and-shoot...just place the cursor over any term on your screen. Chances are we've got it fully detailed in one of our data bases.

Fully cross indexed...if the instruction or library function you're using isn't quite right, our related topics cross index can help you find a better one.

Multiple volumes on line...you can have one or a dozen of our pop-up reference guides on line...a complete library available instantly.

THE INFORMATION YOU NEED...WHERE YOU NEED IT

Our pop-up shell varies its size and shape dynamically, only taking as much space on your screen as it needs and it *never* covers your working area. You can see your work and our reference data at the *same* time.

A COMPLETE LIBRARY...STILL ONLY A BEGINNING

At Santa Rita, our commitment is to provide the most accurate, extensive selection of PC language reference materials available. If you don't see one of our guides for your favorite language or compiler listed below don't worry, we're probably working on it!

PC Programmer's Reference Guide . . . $59.00
(with Assembly Language Guide)

Borland Turbo C *(1.0)* 59.00
Borland Turbo Pascal *(3.0 and below)* 59.00
(with Graphics & Numerical Methods Toolbox)
Borland Turbo Prolog *(1.1 and below)* 59.00
(with Prolog Toolbox)
Lattice C Compiler *(3.2 and below)* 59.00
Mark Williams LetsC *(4.0 and below)* 59.00
Microsoft C Compiler *(5.0 and below)* 59.00

SantaRita

For the location of your nearest Santa Rita Software dealer, or to order direct, call us at 1-214-727-9217. We'd like to hear from you.

Santa Rita Software
1000 E. 14th Street, Suite 365
Plano, Texas 75074

The RESIDENT EXPERT System

Resident Expert is a trademark of The Santa Rita Company. Borland, Turbo C, Turbo Pascal, and Turbo Prolog are trademarks of Borland International Inc. IBM and PC-DOS are trademarks of International Business Machines Corporation. Lattice C is a trademark of Lattice Inc. LetsC is a trademark of Mark Williams Company. Microsoft and MS-DOS are trademarks of Microsoft Corporation.

CIRCLE 4 ON READER SERVICE CARD

FEEDBACK

Artwork: Steve Campbell

Variation on a scheme

Dear Editor:

In "From Turbo Pascal to Assembly and Back" (*COMPUTER LANGUAGE*, Sept. 1987, pp. 51-56), Michael Bertrand used a rather roundabout method of accessing PC registers from within Turbo Pascal. I would like to present a much simpler and clearer method of doing this.

The Turbo Pascal 3.0 manual documents the MS-DOS procedure on page 208. It shows an alternate definition of the PC registers for use in the MS-DOS and *Intr* parameter (Listing 1).

This approach allows you to access whole registers, or just the high or low bytes, without any extra programming. The get-time procedure can be rewritten to reflect this simplification (Listing 2). This approach makes the code easier to understand because it is more similar to the assembly language itself.

It also eliminates the confusion involved when loading AL with zero in order to get the interrupt parameter into AH (in this particlar case, interrupt 21H doesn't check AL for anything, anyway). Also, the *Hi* and *Lo* functions are now unnecessary because the high and low bytes of the registers are accessed directly.

In other programs, this approach eliminates the need to multiply by 256 and the use of *Shl* (shift left) to manipulate registers.

Mike Barnard
Fayetteville, N.C.

Michael Bertrand responds: I'm aware of this variation but didn't include it because variant records can be confusing.

Library closed

Dear Editor:

In their interesting article on Turbo Pascal ("7 Turbo Pascal Idiosyncracies and How to Deal With Them," *COMPUTER LANGUAGE*, Sept. 1987, pp. 59-64), Tom Rugg and Phil Feldman suggested exploring the machine code generated by Turbo Pascal. This task can be made easier by compiling your program as a .CHN file instead of a .COM file. This approach will cause Turbo Pascal to omit the 11K of run-time library, thus producing a smaller file that contains only your program code.

Colin J. Davies
Granada Hills, Calif.

Tom Rugg and Phil Feldman respond: Good suggestion. This approach works well if you have no need to look at that 11K of code generated for Turbo Pascal's internal subroutines. In our experience, however, most programs make frequent calls to these subroutines, and it is often necessary to figure out what they do.

In defense of code

Dear Editor:

As a systems programming manager on a mainframe system for over 10 years, I would add the following comments concerning source code in response to Vince Taylor's letter (Feedback, *COMPUTER LANGUAGE*, Sept. 1987, p. 9).

- In a production shop, the availability of source code for any major software package is almost mandatory.
- When a software bug is encountered at three in the morning or on a weekend, one cannot usually wait until normal working hours to diagnose the problem. The availability of source code can enable an experienced systems support staff to diagnose and, in many cases, solve a serious bug.

```
TYPE RegisterList = RECORD CASE INTEGER OF
1: (AX,BX,CX,DX,BP,DI,SI,DS,ES,Flags: INTEGER
2: (AL,AH,BL,HB,CL,CH,DL,DH: BYTE);
END;

VAR Registers: RegisterList;
```

Listing 1.

```
PROCEDURE Get_Time (VAR Hour, Minute, Second, Hundredth: BYTE);

Type RegisterList + RECORD CASE INTEGER OF
1: (AX,BX CX,DX,BP,DI,SI,DS,ES,Flags:INTEGER);
2: (AL,AH,BL,BH,CL,CH,DL,DH: BYTE);
END;

VAR Registers: RegisterList;

BEGIN
   WITH Registers DO
      BEGIN
         AH := $2C;     { Get the time from DOS }
         MsDos(Registers);  { Make the DOS call (Interrupt 21H) }
         Hour :=CH;
         Minute := CL;
         Second := DH;
         Hundredth := DL;
      END;    {  with  }
 END;  { procedure Get_Time }
```

Listing 2.

Optimizing ROMable C

With Source Level Debugging on Your Target System!

Optimum-C, the compiler known for its tight, highly-optimized code generation, produces completely ROMable code. Coupled with ROM support tools like ROMable standard function libraries, a remote debugger, pre-written startup routine, interactive editor, and UNIX-style MAKE, Optimum-C will help you complete your project in record time.

Optimum-C, the ROM developer's choice

Optimum-C has the features ROM developers require in a C compiler. These features include fast execution, multiple memory support, and 8087 support via inline code or emulation. Optimium-C also has understandable error messages, one-step compiling, complete function prototyping, and inline 8086 I/O instructions.

You've seen the ads: Datalight challenges Microsoft. Our C compiler expert Richard Relph saw the ads and sent for Datalight's compiler. What he found when he began to test it must have given him mixed feelings. For the past two years Richard has been involved in developing the DDJ suite of benchmarks for C compilers. The Datalight compiler flattened those benchmarks, making them worthless.

With the apparent glut of C complier suppliers vying for the MS-DOS market, it was only a matter of time before one of them decided to step above the crowd and provide a reliable optimizing C compiler. Datalight beat all others to the punch by delivering such a compiler February of this year. (Dr. Dobbs Journal, August 1987)

ROM-*it*, The ROM Developer's Kit

ROM-*it* adds the functionality required by serious ROM developers. **ROM-*it*** includes pre-written start-up code for an embedded system to take the 8086 processor from a system RESET to executing C code. The library included has standard C functions that are usable in a ROM environment, without any hidden calls to MS-DOS or the BIOS. Lastly, the BLAZE Locator/Intel hex file generator performs location of code and data while it checks for ROM overflows, illegal data access, and complete program location.

Remote-DSD Source Level debugging on your target system!

The Remote-DSD debugger allows you to debug your application on the target system. Remote-DSD runs on the PC and is connected to your target system via an RS232 link. With Remote-DSD you download your application to the target hardware, modify initial values of variables before you start your application. The C source code appears on the PC showing you where execution will start. You can now execute or single step through your program, set break points, and access your target machine internals like I/O ports, registers, and memory.

And lastly, Support!

The product is only as good as the support. With DATALIGHT, you are covered here. You will get one hour of phone support for those needed answers. Also, you will get updates for one year, so you are always running with the latest versions.

Pick up the phone and gain a powerful advantage.

Call

1-(800) 221-6630

for complete details on how you can finish those ROM projects on time and without the bugs!

Magazine Reviewers Shocked by DATALIGHT's Performance...

"Reviewing this compiler was quite a surprise for us. For such a low price, we were expecting a "lightweight" compiler. What we got was a package that is as good as or better than most of the "heavyweights." Datalight C implements a complete C language. It also compiles quickly, doesn't take up much disk space, and looks impressive in the benchmarks."

DR. DOBBS, August 1986

"This is a sharp compiler!... what is impressive is that Datalight not only stole the compile time show completely, but had the fastest Fibonacci executable time and had excellent object file sizes to boot!"

COMPUTER LANGUAGE, February 1986

Specifications

- Full UNIX V C compiler with ANSI extensions
- Global optimizations using Data Flow Analysis
- 8087/80287 support inline or emulation
- Complete library source code
- Multiple memory model support
- Interrupt handling in C
- Make utility
- EZ editor
- DLC one step compiling
- Start-up code for 8086
- ROMable library (without hidden MS-DOS calls)
- BLAZE locator/Intel hex file generator
- Source level debugging on the target system
- View source code as it executes on target
- Access local variables
- View 8086 machine internals

Datalight

17505-68th Avenue NE, Suite 304
Bothell, Washington 98011 USA
(206) 367-1803

CIRCLE 5 ON READER SERVICE CARD

■ Source code availability enables inserting a vendor's patch with a minimum amount of trouble. Unless a vendor is able or willing to guarantee that updated versions of the product will be made available whenever problems are encountered, source-level patching is the only way to install extensive bug fixes.

■ Most large MIS/DP shops will have unique requirements or workloads that require either functional changes to the software or the tweaking of the software to enhance performance. Most of us understand well the pitfalls of local code and recognize that vendors are under no obligation to fix modified versions of their products. The availability of source code, however, enables us to make these modifications.

I applaud Taylor for making source code available, and wish more vendors did likewise. I have no problem with paying a reasonble fee for either source code or the licence to a vendor's source code. Most of my points are relevant to large MIS/DP shops, but as more vendors try to sell to our market, concerns of this nature are going to have to be addressed.

Richard H. Miller
Head Systems Support
Baylor Computer Center
Houston, Texas

Anarchy (honestly) now!

Dear Editor:

I really enjoyed P.J. Plauger's "Honestly, now," (Programming on Purpose, *COMPUTER LANGUAGE*, Sept. 1987, pp. 13-19). I feel many valid points were made. As a bit of an anarchist myself, however, I feel Plauger sorely misrepresents the political philosophy of anarchy.

Some anarchists, at times, have damaged property, hurt people, and bored others at parties while attempting to install their political philosophies. *No* political philosophies are, to my knowledge, devoid of a few proponents who have tried these techniques. According to *Oxford American Dictionary*, an anarchist is "a person who believes that government and laws are undesirable and should be abolished." From my estimation of some of the affairs in which Plauger was involved, government and laws were far from desirable. The anarchists with whom I have chosen to associate have a high regard for people and the objects they use. ("Property" can imply a legislated status to objects.) Most importantly, some anarchists are a *blast* at parties.

Dale Strickler
McVeytown, Pa.

P.J. Plauger responds: I wasn't referring to the philosophy of anarchy, which aspires to an attractive but unattainable ideal, but to the practice of self-styled anarchists who suffer from the delusion that disrupting society will somehow awaken the populace to an instant appreciation of the anarchist's distress.

Heap o' code

Dear Editor:

First, please allow me something that is more than simply a formality. Your columnist, P.J. Plauger's two texts, his column, and his participation in the X3J11 process I consider to be major contributions to my career, and to our profession. You deserve thanks from all of us!

Not being above the "careful study and imitation of good programs," I was delighted to see a discussion of heap sort in Plauger's "Programming on Purpose," (*COMPUTER LANGUAGE*, Oct. 1987, pp. 17-22). It has served us well! I was reminded of some code in our merge sort: the procedure is named *Reheap* (in keeping with your text's original naming

Professional Programming Tools

For Programmers Compiling in

BASIC • C • COBOL • FORTRAN • PASCAL • dBASE™

GLENCO ENGINEERING is committed to providing professional programming tools. These sub-routines are completely debugged, and fully documented. Hand crafted in ASM, you are assured minimum size and maximum speed. You will literally save hundreds of man-hours in design, program and debug time. All are CALLABLE routines in .OBJ or .COM format. Please specify language and compiler. NOT COPY PROTECTED. NO ROYALTIES.

WINDOWS.LIB* — Create callable pop-up windows, vertical menus, and help screens. Set all borders, colors, size, position. Window editor included. Over 25 commands added to your compiler. $99

WINDOWS with DATA ENTRY* — Option for above. Adds 30 more menu and keyboard enhancement commands. Menu items return integer and text allowing context sensitive help. $99

BIOS.LIB* — Several low level ROM BIOS routines. Categories: KEYBOARD, PRINTER, COMMUNICATIONS, VIDEO and MISC. $99

DOS Functions* — Execute another program or alter DOS internal switches from inside your program. Read or set DTA, Interrupt Addrs., PSP, Verify, MEM, default drive, I/O and more . . . $99

FILES LIB* — Allows virtually any DOS file operation from inside your program. Read or set a files' Attribute, Mode, Data, Time. Check Write protect tab, free space, dir. $99

NETWORK TOOLS — Over 80 subroutines to interface to NETBIOS. Transaction process, multitasking, communications, and more . . . Works with any system supporting NETBIOS, or eqvt. $99

*Buy 4 get 5 or Library 6 Pack - $450
Multiple purchases discount and site licensing.

NO ROYALTIES

• IBM™
• Summit Software™
• Computer Innovations™
• Microsoft™
• Lattice™
• Ryan McFarland™
• Realia™
• Nantucket™
• Borland™
• Aztec
• Wordtech™
• and More

Call or Write:
GLENCO ENGINEERING INC.
SERVING THE SOFTWARE INDUSTRY SINCE 1979
3920 Ridge, Arlington Hts., IL 60004 (312) 392-2492 Telex 493-7109

VISA MasterCard

CIRCLE 6 ON READER SERVICE CARD

The Leaders Made PVCS The Leading Source Code Control System.

NOW VAX/VMS & MS-DOS Versions

When it comes to maintaining their most valuable asset, the leading software publishers rely on the POLYTRON Version Control System (PVCS). From accounting firms to airlines, the leading service companies depend on PVCS to maintain the integrity of their programs. Leading manufacturing companies use PVCS to maintain their state-of-the-art software. Leading high technology companies turn to PVCS to handle configuration management for software projects that represent an investment of hundreds of thousands of dollars. The largest aerospace companies and defense contractors use PVCS to maintain integrity of projects during development and after delivery of software. Independent programmers use PVCS to improve their productivity and software quality for themselves and their clients.

Simplify Configuration Management

When large and complex software programs are being developed on personal computers or VAX minicomputers, effective management of the revisions and versions becomes critical. PVCS simplifies this process and lets you effectively control the proliferation of code changes. We used UNIX SCCS and RCS as models. However, our own experience, and the input of hundreds of programmers and managers has enabled us to significantly improve upon these models.

PVCS provides many powerful functions including:

- Storage & Retrieval of multiple revisions of text.
- Maintenance of a complete history of changes.
- Maintenance of separate lines of development using branching.
- Merging simultaneous changes.
- Resolution of Access Conflicts.
- Modules can be retrieved by their own revision number, system version name, or specified date.
- Uses "reverse deltas" to rebuild a prior version making PVCS the fastest version control system over the project life cycle.
- Projects already under development or in the maintenance stage can be easily put under the control of PVCS.

Manages Development On Local Area Networks

Programming teams using Local Area Networks depend on PVCS to help the managers and team members work together. In fact, Novell and 3Com themselves depend on PVCS to manage the versions of their own network software products.

Supports MS-DOS and VAX/VMS Development

Now, companies that develop software on VAX systems running VMS can also use PVCS. And since the VMS and MS-DOS versions of PVCS use the same "logfile" format, you can easily develop software on PCs and maintain the code on the VAX or vice versa. The menu-driven, screen-oriented interface (and optional command-driven interface) makes it easy for programmers and librarians or administrators to use PVCS on a PC or VAX or both systems.

PVCS Maintains System Integrity

PVCS prevents corruption of code that could ordinarily result from security breaks, user carelessness or malfunctions. The levels of security can be tailored to meet the needs of your project.

PVCS & PolyMake Work Together

PolyMake, the leading MS-DOS make utility, is now available for the VMS operating system. This allows you to write makefiles that will function in both PC and VAX environments. Additionally, PolyMake reads time & date stamps of PVCS archives for fast, accurate program rebuilding.

PVCS and PolyMake Maintain Source Code Written In Any Language.

The Preferred Version Control System

The customers listed below are just a few of the innovative leaders that have made PVCS the leading version control program for personal computers.

Alcoa Aluminum
Arthur Anderson
AT&T
Ashton-Tate
Bank of America
Bell Labs
Bendix
Boeing
CIGNA
Citibank
3Com
Colonial Penn
Commerce Clearing House
Control Data Corp.
Corvus
CXI
Digital Equipment Corp.
Deloitte Haskins + Sells
Diebold
Dow
Dunn & Bradstreet
EDS
Educational Testing Service
E-Systems
Equitable Life
Federal Express
First Boston
Ford
Fox Software
Fujitsu
GTE
Hardees
Hewlett-Packard
Honeywell
Hughes Aircraft
IBM
Industrial Networking
Intel
ISC Aerospace
IVAC
Javelin
Lattice
Lawrence Livermore
Lotus
McData Corp.
McDonnell Douglas
Mead Data Central
MIT Lincoln Labs
Nastec
Novell
NCR Technologies
Pitney Bowes
Plexus Computers
Price Waterhouse
ROLM
Rockwell International
Safeco
Sears
Security Pacific
Sperry
Software Publishing
Spacelabs
Standard Oil
Standard & Poors
Tandem
Tektronix
Telex
Texas Instruments
Touche Ross
Unisys
United Airlines
United Parcel Service
United Technologies
U.S. West
Westinghouse Electronics
Xerox

Only PVCS meets the needs of independent programmers and corporations. Once you standardize on PVCS, the archives used to track and monitor changes are interchangeable between any PVCS product. You will receive full credit for your initial purchase if you upgrade to a higher-priced MS-DOS version of PVCS.

Personal PVCS – Offers most of the power and flexibility of Corporate PVCS, but excludes the features necessary for multiple-programmer projects.

Corporate PVCS – Offers additional features to maintain source code of very large and complex projects that may involve multiple programmers. Includes multi-level branching to effectively maintain code when programs evolve on multiple paths (e.g. new versions for different host systems, or a new program based on an existing program).

Network PVCS – Extends Corporate PVCS for use on Networks. File locking and security levels can be tailored for each project.

PVCS for VAX systems – Requires VMS. Uses the same interface and archive format as MS-DOS version. Supports branching and offers file locking and other security features for multiple-programmer projects.

	MS-DOS*	VMS		
	PC/XT/AT	Micro VAX II	VAX 7xx	VAX 8xxx
Personal PVCS	$149			
Corporate PVCS	$395			
Network PVCS	$995**	$4,950	$9,500	$10,500+
PolyMake	$149			
Network PolyMake	$447**	$1,250	$2,375	$2,500+

*Compatible with MS-DOS 2.0 through 3.3. Compatible with the IBM PC/XT/AT & other MS-DOS PCs.

**5 Station LAN License. Call for pricing on larger Networks.

TO ORDER:
VISA/MC 1-800-547-4000.
Dept. No. 350.
Oregon & Outside USA call (503) 645-1150.
Send Checks, P.O.s to: POLYTRON Corporation, 1815 NW 169th Place, Suite 2110, Beaverton, OR 97006.

POLYTRON
High Quality Software Since 1982

CIRCLE 7 ON READER SERVICE CARD

scheme). However, the comment on *our* procedure reads "Actually, just puts the smallest one first." I'm not sure I should be bragging about the discrepancy between our procedure's name and operation, though. However, I do want to point out to you that (as far as I can equivicatingly tell) there is no need to do a heap sort, however elegant it may be. As our code's comment described, the routine need only return the *heap* with the first element being its smallest. This, of course, makes for smaller, faster, and less buggy code, eh?

Tom Field
Quickpen International
Englewood, Col.

Adding to the repertoire

Dear Editor:

Thank you for your review of PMI's Repertoire (Software Reviews, *COMPUTER LANGUAGE*, Sept. 1987, pp. 118-119). An additional point and a minor correction ought to be made, however.

A month or two after Warren Keuffel finished his review, Logitech began shipping a remarkable new linker with Modula-2/86 v. 3.0. Unlike most other DOS linkers, it is capable of linking in only as much code as a program actually references, regardless of the extent to which the library is granularized into separate modules. As a result, Repertoire's enormous size and tight integration no longer cause needless code to be linked in; an application incorporates only as much code from a library as it really executes.

Thus Repertoire's list management and indexed file systems do, in fact, work with structured records. If the reserved word *record* is used as the field name, the file system can store and retrieve true binary images of any object, including structured records, arrays, real numers, bit maps, etc.

We realize that this feature is not well documented in the present release. Keuffel correctly described the primary purpose of Repertoire's indexed file system as the storage of complex variable-length and list oriented data.

Cole Brecheen
Vice president, PMI

A paradoxical exclusion

Dear Editor:

It was interesting to read Richard Finkelstein's "Wrap-up rebuttal" (Feedback,

A Different BASIC Might Make All the Difference

We'll skip the four-color gatefold. And the extravagant claims. Because if you're serious about programming, you just want the straight facts:

	True BASIC 2.01	Microsoft Quick Basic 3.0	Borland Turbo Basic 2.0
GRAPHICS			
Supports Hercules Graphics	YES	NO	NO
Device-Independent Graphics Syntax	YES	NO	NO
User-Defined Coordinates	YES	LIMITED	LIMITED
Matrix Graphics Coordinates	YES	NO	NO
ARRAY HANDLING			
Matrix Algebra	YES	NO	NO
Maximum Numeric Array	UNLIMITED	64K	64K
Max. Number of Array Dimensions	255	63	8
Max. Number of Elements/Dimension	UNLIMITED	32K	32K
Dynamic Redimensioning	YES	NO	NO
Matrix I/O Statements	YES	NO	NO
STRING/FILE HANDLING			
Maximum String Length	64K	32K	32K
Total String Space	UNLIMITED	64K	64K
Maximum Record Size	16MB	32K	32K
Max. Bytes/Binary File Read	64K	NA	32K
PRODUCTIVITY ENHANCERS			
Modules	YES	NO	NO
Separately Compiled Libraries	YES	LIMITED	NO
Workspaces	YES	NO	NO
Immediate Mode	YES	NO	NO
SPECIAL FEATURES			
Stop/Continue Execution	YES	NO	NO
Max. Source File	UNLIMITED	UNLIMITED	64K
Script Files	YES	NO	NO
Keystroke Macros	YES	NO	NO
Max. Characters/Line	64K	255 char.	249 char.
Max. Scalar Data Space	UNLIMITED	64K	64K
Mouse Support	YES	NO	NO
80386 Version	YES	NO	NO
Portability to:	Macintosh, Amiga, Atari	Translation required	No other machines

NOW ONLY $99.95

Three very structured, very powerful programming packages. All with fancy editors and fast compilers. Two of them are the same in other respects. And one of them, True BASIC, is quite a bit different. With syntax and features that will make you more productive.

That's why reviewers for magazines like BYTE, PC Tech Journal and Computerworld keep giving True BASIC their top marks. And why OEMs pick True BASIC after they've evaluated all the others. See why True BASIC can make the difference for you.

Call **1-800-TRBASIC** today.

True BASIC

True BASIC, Quick Basic and Turbo Basic are trademarks of True BASIC, Inc, Microsoft and Borland, respectively. Macintosh, Amiga and Atari are trademarks of Apple Computer, Inc., Commodore-Amiga, Inc. and Atari Corporation. Copyright 1987 True BASIC. Specifications are accurate as of August 1987.

39 SOUTH MAIN STREET
HANOVER, N.H. 03755
(603) 643-3882

CIRCLE 8 ON READER SERVICE CARD

Remove the Limitations of .BAT Files with a Superior Extended Batch Language . . .

"I hit a wall early on with .BAT files. OPAL has infinitely expanded my batch capabilities" ... Ron Fluegge, Texas Utilities Electric Co.

We all know PC/MS-DOS BATCH is limited---all you get are a few commands. That's why you need OPAL.

OPAL provides an interpretive language that makes it easy to program the control you want in your applications. OPAL has a comfortable, solid set of commands that provide you a flexible shell offering a high degree of programming, including flow-of-control, menus and forms, CALLs, DO groups, etc. Functions are available for dates, strings, directories, files, numerics, system, and console, to name a few.

With OPAL you get a complete environment with options for a screen painter, library manager, Math Option, and more. And, once your application is built, if you want to optimize and securely package your software, OPAL's Runtime System is ready and waiting.

Make a quantum leap past DOS BATCH limitations---order OPAL today.

Special offer through Dec. 15

.....only $99

Texas residents add 7.75% sales tax
Shipping $4.00 (US) $12.00 (outside US)
Mastercard/VISA accepted

To order or receive more information:

(214) 490-0835

The Software Factory
INCORPORATED
15301 Dallas Parkway
Suite 750 LB 44
Dallas, TX 75248

IBM is a registered trademark of International Business Machines Corp.
OPAL is a trademark of The Software Factory, Inc.

CIRCLE 10 ON READER SERVICE CARD

You have been *waiting* for this!

Turbo Instant_Compile T.M.

A new utility for Turbo Pascal programmers.

Drastically reduces the tedious and repetitive recompilation of previously debugged source code.

Ideal for programmers who appreciate Turbo's programming environment but who:

- Tire of watching Turbo's line numbers flash by the screen
- Fail, occasionally, to get everything entirely right the first time
- Use a library of purchased or previously developed code
- Believe computers should be adapted to serve humans

Instant_Compile saves your time and sanity by eliminating the need to repetitively compile perfectly good code in order to reach the portion you are developing. It is easy to use. You place a compiler directive in the source code. When this point is reached, the entire state of the compilation is saved. Your next compilation can ***instantly*** return to this point and proceed. Simple to use.

Now it is practical to use Turbo Pascal in developing larger programs. Requires: 256K IBM PC or compatible and Turbo Pascal 3.01A or 3.02A. Why wait? Call now!

(817) 654-5400

Introductory price: Only $49

100% satisfaction or your money back.

Synergy Systems T.M.

200 West 7th St. #613 ● Ft. Worth, Texas 76102

CIRCLE 9 ON READER SERVICE CARD

COMPUTER LANGUAGE, July 1987, p. 11). Even though I am not an employee of ANSA Software or heavy user of Paradox I found his justification for its exclusion from his review to be non-informed.

For example, he gives as one of his reasons the fact that the reviewers chose to review only nonprocedural, set-oriented products. But in the original review (Software Reviews, *COMPUTER LANGUAGE*, May 1987, pp. 89-116), one of the products reviewed was PROGRESS which, it was stated, has no nonprocedural way to build screen entry forms. He then gave an example of highly procedural PROGRESS code to show how table updates might be done. In PARADOX, the applications generator allows completely nonprocedural specification of an entire application, screens included. You would never need to write such code.

In addition, I find it odd for him to say that a package with SQL has anything going for it over the QBE used by Paradox, because the underlying techniques and logic required to execute a QBE query are identical to those needed to execute SQL, with the exception of the nested SQL query blocks.

QBE queries are as powerful as any equivalent to the active set of records returned by an SQL query. And PAL, the applications language for Paradox is at least as nonprocedural as any other programming langage with embedded SQL, with the exception of something like Informix/4GL. But with Paradox, the Apgen is so powerful, you may never need to write any code of any generation, which is about as nonprocedural as most of us could hope for.

Gary T. Lang
Ratliff Software Productions Inc.
Diamond Bar, Calif.

Richard Finkelstein responds: As I mentioned in my review, PROGRESS has some procedurality. Better examples of nonprocedural SQL development systems include XDB, Oracle, Ingres, and Informix, the highest rated products in the annual data base derby competition, which measures nonprocedurality. Unfortunately Paradox declined to enter the derby last year. Paradox as a PAL language appears to be very similar to BASIC, with the exception that PAL allows for QBE statements. QBE is a subset of SQL and cannot perform many equivalent SQL functions, including relational division operations. Hundreds of application generators for procedural languages are available, many of them equal to or surpassing Paradox. Since developers have to enhance, maintain, and debug the object code of these generators, it is important that the object code be nonprocedural. PAL is not.

Our thanks to NASA for supplying this computer enhanced ultraviolet photo taken by Skylab IV of a solar prominence reaching out 350,000 miles above the sun's surface

Genius Begins With A Great Idea ...

But The Idea Is Just The Beginning

What follows is the time consuming task of giving form and function to the idea.

That's why we concentrate on building into our software development systems functions and features that help you develop your software ideas in less time and with less effort.

We've started 1987 by releasing new versions of our MS-DOS, Macintosh, Amiga, ROM, and Apple // C development systems. Each system is packed with new features, impressive performance, and a little bit more genius.

FREE 2-DAY DELIVERY

Aztec C86 4.1

New PC/MS-DOS • CP/M-86 • ROM

Superior performance, a powerful new array of features and utilities, and pricing that is unmatched make the new Aztec C86 the first choice of serious software developers.

Aztec C86-p Professional System $199
• optimized C with near, far, huge, small, and large memory + Inline assembler + Inline 8087/80287 + ANSI support + Fast Float (32 bit) + optimization options • Manx Aztec 8086/80x86 macro assembler • Aztec overlay linker (large/small model) • source level debugger • object librarian • 3.x file sharing & locking • comprehensive libraries of UNIX, DOS, Screen, Graphics, and special run time routines.

Aztec C86-d Developer System $299
• includes all of Aztec C86-p • Unix utilities make, diff, grep • vi editor • 6+ memory models • Profiler.

Aztec C86-c Commercial System. $499
• includes all of Aztec C86-d • Source for library routines • ROM Support • CP/M-86 support • One year of updates.

Aztec C86 Third Party Software

A large array of support software is available for Aztec C86. Call or write for information. The following is a list of the most requested products: • Essential Graphics • C Utility Library • Curses • Greenleaf Communication, General, and Data Window • Halo • Panel+ • PC-lint • PforCe • Pre-C • Windows for C • Windows for Data • C terp • db_Vista • db-Query • Phact • Plink-86 Plus • c-tree • r-tree • Pmate

CP/M • TRS-80 • 8080/Z80 ROM

C compiler, 8080/Z80 assembler, linker, librarian, UNIX libraries, and specialized utilities.

Aztec C II-c (CP/M-80 & ROM). $349
Aztec CII-d (CP/M-80) $199
Aztec C80 (TRS-80 3&4) $199

Aztec C68k/Am 3.4

New Amiga Release

Amiga user groups across the USA voted Aztec C68k/Am release 3.3 the best Software Development System for the Amiga. Release 3.4 is more impressive.

Aztec C68k/Am-p Professional $199
A price/feature/performance miracle. System includes: optimized C • 68000/680x0 assembler • 68881 support • overlay linker • UNIX and Amiga libraries • examples.

Aztec C68k/Am-d Developer $299
The best of Manx, Amiga, and UNIX. System includes: all of Aztec C68k/Am-p • the Unix utilities make, diff, grep and vi.

Aztec C68k/Am-c Commercial $499
Aztec C68k/Am-d plus source for the libraries and one year of updates.

FREE 2-DAY DELIVERY

Aztec C68k/Mac 3.4

New Macintosh Release

For code quality, reliability, and solid professional features, Aztec C for the Macintosh is unbeatable. This new release includes features and functions not found in any other Macintosh C development system.

Aztec C68k/Mac-p Professional $199
• MPW source level compatibility • TMON, MACSBUG, AND MACNOSY support • powerful symbolic debugger • optimized C • 68000/680×0 assembler • 68881, EEE, and SANE support • overlay linker • UNIX and Macintosh libraries • mouse editor • examples

Aztec C68k/Mac-d Developer. $299
The best of Manx, Macintosh, and UNIX. System includes: all of Aztec C68k/Am-p • the Unix utilities make, diff, grep • vi editor.

Aztec C68k/Mac-c Commercial $499
Aztec C68k/Am-d plus source for the libraries and one year of updates.

Aztec C65 — New ProDOS Release

Aztec C65 is the only commercial quality C compiler for the Apple II. Aztec C65 includes C compiler, 6502/65C02 assembler, linker, library utility, UNIX libraries, special purpose libraries, shell development environment, and more. An impressive system.

Aztec C65-c Commercial $299
• runs under ProDOS • code for ProDOS or DOS 3.3

Aztec C65-d Developer. $199
• runs under DOS 3.3 • code for DOS 3.3

Aztec ROM Systems

6502/65C02 • 8080/Z80 • 8086/80x86 • 680x0

An IBM or Macintosh is not only a less expensive way to develop ROM code, it's better. Targets include the 6502/65C02, 8080/Z80, 8086/80x86, and 680x0.

Aztec C has an excellent reputation for producing compact high performance code. Our systems for under $1,000 outperform systems priced at over $10,000.

Initial Host Plus Target $750
Additional Targets $500
ROM Support Package $500

Vax, Sun, PDP-11 ROM HOSTS

Call for information on Vax, PDP-11, Sun and other host environments.

C' Prime

PC/MS-DOS • Macintosh
Apple II • TRS-80 • CP/M

These C development systems are unbeatable for the price. They are earlier versions of Aztec C that originally sold for as much as $500. Each system includes C compiler, assembler, linker, librarian, UNIX routines, and more. Special discounts are available for use as course material.

C' Prime $75

Aztec Cross Development Systems

Most Aztec C systems are available as cross development systems. Hosts include: PC/MS-DOS, Macintosh, CP/M, Vax, PDP-11, Sun, and others. Call for information and pricing.

How To Become An Aztec C User

To become a user call 800-221-0440. From NJ or international locations call 201-542-2121. Telex: 4995812 or FAX: 201-542-8386. C.O.D., VISA, MasterCard, American Express, wire (domestic and international), and terms are available. One and two day delivery available for all domestic and most international destinations.

Aztec C is available directly from Manx and from technically oriented computer and software stores. Aztec Systems bought directly from Manx have a 30 day satisfaction guarantee.

Most systems are upgradable by paying the difference in price plus $10. Site licenses, OEM, educational, and multiple copy discounts are available.

MANX

Manx Software Systems
1 Industrial Way, Eatontown, NJ 07724

To order or for more information call today.

1-800-221-0440

In NJ or international call (201) 542-2121 • TELEX: 4995812

MS is a registered TM of Microsoft, Inc., CP/M TM DRI, HALO TM Media Cybernetics, PANEL TM Roundhill Computer Systems, Ltd., PHACT TM PHACT Assoc., PRE-C, Plink-86, Plink-86+, P-Force TM Phoenix, db Vista TM Raima Corp., C-terp, PC-lint, TM Gimpel Software, C-tree TM Faircom, Inc., Windows for C, Windows for DATA TM Creative Solutions, Apple II, Macintosh TM Apple, Inc., TRS-80 TM Radio Shack, Amiga TM Commodore Int'l., Unix TM AT&T, Vax TM DEC, Aztec TM Manx Software Systems.

CIRCLE 11 ON READER SERVICE CARD

SAVE YOUR PC FROM EARLY RETIREMENT.

GET HAUPPAUGE'S NEW 386 MOTHERBOARD.

386 SPEED–ONLY $1,495

Give your PC a new lease on life! With our industry first 386 MotherBoard, your PC, PC/XT or compatible will revel in speeds equal to the Compaq DeskPRO 386. And faster. Because we've built in 1 Megabyte of high speed RAM and a 387 math coprocessor socket for speeds that will knock you off your rocker.

To keep retirement at bay, our 386 MotherBoard is compatible with the PC/AT (BIOS and I/O) — allowing you to run the new generation of DOS, OS/2. We've also included a 16-bit expansion slot that accommodates the latest I/O expansion card. No accelerator card can give you so much versatility.

With 386 power and true AT software compatibility, your business, desktop publishing and engineering applications will get a boost to boast about! **Technical Features** ▪ 16 MHz 80386 ▪ 1 Megabyte of 100 nsec 4-way interleaved RAM ▪ PC/AT compatible I/O and BIOS for support of OS/2 ▪ Six 8-bit expansion slots ▪ Two 16-bit expansion slots ▪ One 32-bit RAM expansion slot ▪ Optional 16 MHz 80387 math coprocessor ($695)

Put the power of the 386 into your IBM PC for 1/4 the cost of a 386 computer. And put off your PC's retirement. For more information on our easy-to-install Motherboard, call **1 (800) 443-6284.** In New York, call **(516) 360-3827.**

Hauppauge Computer Works, Inc.
358 Veterans Memorial Highway,
Commack, New York 11725

Hauppauge!

Trademarks: IBM PC/AT and PC/XT IBM, DeskPRO 386, Compaq Computers

CIRCLE 12 ON READER SERVICE CARD

PROGRAMMING ON PURPOSE

Synchronization in hard-to-easy design

By P.J. Plauger

Method.
Hard-to-easy design.

Description.
Hard-to-easy design focuses on safe cooperation among sequential processes. When independent agents endeavor to share data, synchronization is required to prevent race conditions and data corruption. When synchronization is poorly implemented, lapses and deadlock may occur.

The name derives from the observation that synchronization is hard to get right and best dealt with early on in the design process. The primitives used to effect synchronization vary widely among different systems. Safe operation often can be achieved only at the cost of overall system performance. Thus, synchronization requirements should be kept to a bare minimum.

Documentation.
Timing diagrams show critical transitions that assure proper synchronization.

Limitations.
You cannot always separate synchronization issues from background computation, particularly in real-time systems with sophisticated feedback. It is hard to avoid all possibility of deadlock in a complex system.

Once upon a time, computers were fairly simple. When your program wanted to read a card, it issued a *READ* instruction to the card reader, then proceeded to twiddle its thumbs. Nothing else happened while the picker separated the bottom card from the deck in the input hopper, the feed mechanism dragged the card past the read heads, and the stacker dropped it neatly in the output hopper.

A very large fraction of a second after your program issued the *READ* instruction, it moved on to the next instruction, secure in the knowledge that the data from the card was copied into the designated buffer in memory. Even in those unhurried times programmers lamented the milliseconds wasted while bits of matter were coaxed into motion by various peripheral devices.

So computer designers developed the input/output channel, a moderately stupid processor that knew just enough to supervise the transfer of data to or from a peripheral device. The central processing unit issued its *READ* request to the I/O channel, then went on about its business. When a transfer was complete, the I/O channel jerked the chain of the CPU by sending an interrupt. Your program could then note, in some fashion, that the transfer was complete, and liberate the buffer in memory for other uses.

At that point the systems programmers made a mistake. They assumed that the control of overlapped I/O should be put into the hands of every applications programmer. Suddenly, FORTRAN programmers had to deal with new I/O options of frightening power. We had to specify event flags to be set on successful completion of a transfer. We had to specify handlers to be entered on transfer error, end of file, and possibly even successful completion. Most importantly, we had to learn to keep our cotton-picking hands off the I/O buffer until each transfer was complete.

Synchronization is hard to get right and best dealt with early on in the design process

If we didn't take on all this baggage, we suffered the stigma of using the computer inefficiently (a social faux pas roughly equivalent to turning down a date with Brooke Shields). If we did take it on, we were forced through a programmed learning course in how to nominally synchronize cooperating sequential processes—with no textbook for the course and our application program providing the only courseware.

Eventually, the systems programmers took pity on FORTRAN programmers. They provided spoolers that used overlapped I/O to read ahead on card readers and write behind on printers and punches. We went back to writing programs that completed each transfer before proceeding, certain that most transfers involved simply a memory-to-memory buffer copy. Computer utilization was up and we were in a state of social grace. Only the macho types persisted in coding their own overlapped I/O.

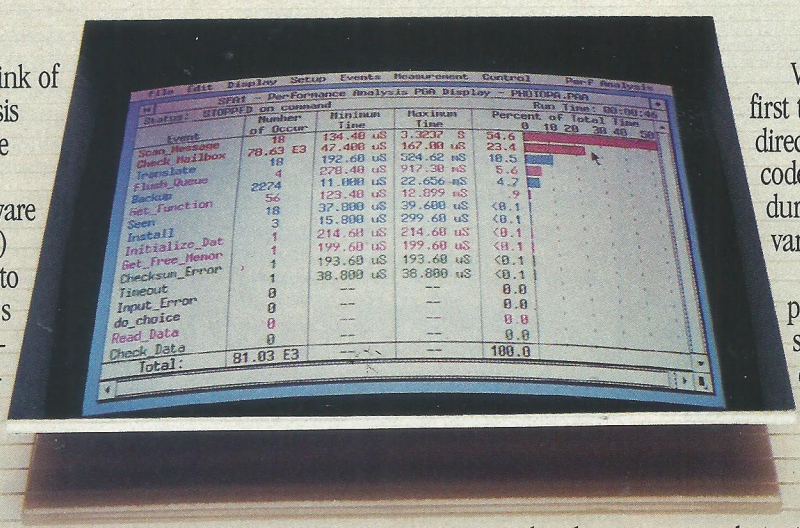
Just as that flap was settling down, another one emerged. Computers were being equipped with multiple terminals, allowing several people to simultaneously negotiate with separate programs, all running on the same machine. Conversational time-sharing was invented as a way of keeping people seated while waiting for better response time instead of milling about the ready room waiting for spooled printer output.

To make all this happen, systems programmers developed operating systems that could juggle multiple processes or separate threads of control. (Think of a process as a program that thinks it has a whole machine to itself, even though it is a virtual machine that is just a subset of the underlying hardware.)

At that point the systems programmers made another mistake. They assumed that the ability to run multiple processes should be put into the hands of every applications programmer. Suddenly, PL/I programmers had to deal with tasking options of frightening power. We had to specify event flags to be set on successful completion of a subtask. We had to specify handlers to be entered on tasking errors. Most importantly, we had to learn to keep our cotton-picking hands off any storage used by another task until its usage was assuredly complete.

Not using tasking never quite attained the stigma associated with not using overlapped I/O. What was worse, overlapped I/O looked easy to use. Nowhere in the PL/I manual was the least hint that tasks were hard to specify safely, expensive to start up, or next to impossible

Everything you know about performance analysis is wrong.



For starters, you probably think of software performance analysis tools as some sort of software package.

Wrong. The NWIS Software Analysis Workstation™ (SAW) uses sophisticated hardware to sample your software. There's no intrusion into your memory space or CPU cycle stealing. The SAW operates in real-time and non-statistically. You capture *every* occurrence of the events specified, not just a random sample.

You might also think performance analysis is limited to very narrow windows of real-time execution.

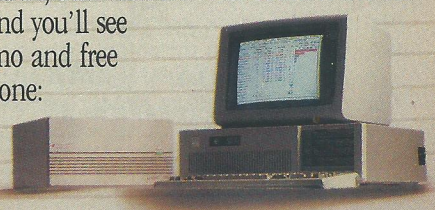
Wrong again. With the SAW, you can now analyze days of execution in a single run, instead of just a millisecond or two. With total accuracy.

You also might associate performance analysis with assembly code. Even though the majority of your code is probably compiled from high-level source.

Very wrong. The SAW is the first tool which lets you work directly with high-level source code definitions, such as procedures, modules and global variables.

Finally, you might imagine performance analysis to be a stand-alone tool. But the SAW changes that for good. In the very same system, you get high-level symbolic trace, assembly-level trace, time-aligned multi-processor trace and code coverage analysis. All these tools are compiler/assembler and host development computer independent.

In short, the SAW is everything you hoped software performance analysis could be, but wasn't. Let us connect the SAW to your target system and you'll see why. For an on-site demo and free applications library, phone: 1-800-547-4445



NWIS

NORTHWEST INSTRUMENT SYSTEMS, INC.

P.O. Box 1309 • Beaverton, OR 97075 • 1-800-547-4445

CIRCLE 13 ON READER SERVICE CARD

AUSTRIA: 0222-253626 **BELGIUM**: 02-2418130 **DENMARK**: 05-611100 **FINLAND**: 0-5284316 **FRANCE**: 014-5347535 or 016-9070935 **ISRAEL**: 03-403373 **ITALY**: 02-470646 **THE NETHERLANDS**: 01720-43221 **NORWAY**: 02-789460 **SPAIN**: 01-4558112 **SWEDEN**: 08-7390045 **SWITZERLAND**: 01-7231410 **UNITED KINGDOM**: 06284-4426 **NWIS-USA**: 503-690-1300 **WEST GERMANY**: 089-858020

to debug. Once again, the tuition we paid did not entitle us to a textbook.

Eventually, the systems programmers again took pity on us. They provided command interpreters that start and administer multiple processes on our behalf. They gave us processes that run on separate virtual machines and communicate only through the operating system.

They invented pipelines and first-in-first-out (FIFO) files for sequential data passing. They invented record locking for shared access to files. By eliminating most of our freedoms, they set us free. Only the macho types persisted in writing unstructured, multithread programs.

I have just described what is generally regarded as the first three generations of computers. (The distinction between generations has since dissolved in an acid sea of marketing hype.) In summary, these are:

- First generation—a single thread of control in a private memory
- Second generation—overlapped I/O, used to improve the performance of a single thread of control in a private memory
- Third generation—multiprocessing used to support largely independent processes, each a single thread of control in a private memory.

The point I am belaboring is that using overlapping multiple operations to work toward a common goal is a great way to improve the performance of a computer system but is very hard to control. Synchronizing logic is the hardest code in the world to design, debug, and tune. Once you get it right, you stuff it in a black box, hold it at arm's length, and treat it as just another peripheral connected to your first generation virtual machine.

Ninety-odd percent of the code you write should be written for a single-thread environment. Otherwise, you'll pay a stiff premium on everything you write. This is why I call writing synchronizing code hard-to-easy design. If you have to do it, do it first, before the problem gets cluttered with more mundane details. Identify where cooperating sequential processes absolutely must share information to cooperate and keep that interface to a bare minimum. Then encapsulate that interface, both data and synchronizing functions, and clearly document the rules for using it. Finally, debug the synchronization code alone as best you can before you go on.

This fear of synchronization code may appear unseemly. All I can say is that of all the designing and debugging I have ever done, none has come close to presenting the same level of difficulty.

With a complex program you develop a feel for how many changes you can make before losing control of the program. I have found that an operating system kernel that is the same size as a language translator can safely tolerate only about a fifth as many changes as the translator before it breaks. And when an operating system breaks, it quickly turns into an uncooperative lump that is inert at best and insanely malicious at worst.

If you are now sufficiently frightened, let's look at what it takes to synchronize multiple activities. Once again, I will recapitulate (more or less) the evolution of multithread systems from the simplest to the most ambitious.

Say you need to design a microprocessor-based system that gathers data from a number of sources. The easiest way is to provide a simple hardware interface for each of these sources that holds onto the most recent data until the computer can do something with it.

From time to time, your program examines a status bit in the control registers for each of the hardware interfaces. If the bit is set, your program knows that data is present. It reads the data and clears the control bit to inform the hardware interface that the data has been delivered. This matters only if your program fails to collect a piece of data before the next one comes along.

At that point the hardware interface suffers a data overrun, which it should report to your program via yet another status bit. Your program is obliged to do something sensible in the presence of data overruns. The following is an example of the control code:

```
FOREVER
   IF (<data overrun>)
      <evince concern>
      <clear overrun>
   IF (<data ready>)
      item := <input data>
      <clear data ready>
      process(item)
```

This method is popular for interfacing to computers, particularly microprocessor-based systems, where cost and complexity must be minimal. It requires a style of programming called polling. The design centers around an infinite loop that polls each device in turn, looking for more work to do.

Synchronization is simple because the control program gathers data only when it is ready for it, then completely disposes of the data before assuming the next task. Provided that system performance is high enough to keep data overruns to a minimum and that such data overruns are not cataclysmic, the solution has ad-

FULL AT&T C++ for half the price of our competitors!

Guidelines announces its port of **version 1.1** of AT&T's C++ translator. As an object-oriented language, C++ includes: classes, inheritance, member functions, constructors and destructors, data hiding, and data abstraction. 'Object-oriented' means that C++ code is more readable, more reliable and more reusable. And that means faster development, easier maintenance, and the ability to handle more complex projects. C++ is **Bell Labs' answer to Ada and Modula 2.** C++ will more than pay for itself in saved development time on your next project.

C++

from GUIDELINES for the IBM PC: $195

Requires IBM PC/XT/AT or compatible with 640K and a hard disk.
Note: C++ is a *translator*, and requires the use of Microsoft C 3.0 or later.

Here is what you get for $195:
- The full AT&T v1.1 C++ translator.
- Libraries for stream I/O and complex math.
- "The C++ Programming Language", the definitive 327-page tutorial and description by Bjarne Stroustrup, designer of C++.
- Sample programs written in C++.
- Installation guide and documentation.
- 30 day money back guarantee.

To order:
send check or money order to:
GUIDELINES SOFTWARE
Dept. CL
P.O. Box 749
Orinda, CA 94563
To order with Visa or MC, phone (415) 254-9393.
(CA residents add 6% tax.)

C++ is ported to the PC by Guidelines under license from AT&T.
Call or write for a free C++ information package.

CIRCLE 14 ON READER SERVICE CARD

- *The* **KORN** *Shell* -

KSH

At last! The genuine AT&T **KORN** Shell is available for **XENIX®** and **UNIX®** System V operating systems. This is a direct port of the popular UNIX® System **KSH** command interpreter used throughout AT&T and Bell Laboratories, not a re-write!

AVAILABLE FOR:
XENIX System V on AT & Compatibles
MICROPORT System V/AT
AT&T 3B2 Computers with UNIX System V

Other machines are in the works - Call us.

— **Bourne Shell users!**, **KSH** allows re-entry of previous commands and in-line editing. Command aliasing gives you useful short-hand access to complex commands.

— **C Shell users!**, Many of the facilities you like in Csh are available in **KSH**: Command History, Integer Arithmetic, and Arrays. **KSH** provides all this and more in a Bourne compatible interpreter.

— **Shell Programmers!**, **KSH** provides the power of the Bourne Shell and adds Arithmetic and String Evaluation, Local Variables, and additional I/O capabilities.

A GREAT VALUE!

Xenix System V - **$125.00**
Microport System V - **$125.00**
AT&T 3B2 - **$195.00**

Also available,
UNIX Shell Programming by *Kochan & Wood;* contains excellent chapter on **KSH** programming — **$19.00**

For more information or to order, write:

ASPEN TECHNOLOGIES, INC.
P. O. Box 5727
Parsippany, NJ, 07054

or call:
Collect (201) 316-0866

Check or Money Order, **VISA** and **MASTERCARD** accepted. Include $3.00 for Shipping & Handling (NJ Residents add 6% Sales Tax).

Volume discounts available.
Dealer and OEM inquiries invited.

KSH - The KORN Shell

XENIX is a registered trademark of Microsoft Corporation, UNIX is a registered trademark of AT&T, IBM and PC-AT are registered trademarks of International Business Machines Corporation. TANDY is a registered trademark of Tandy Corporation. KSH is based upon software in the AT&T UNIX System Toolchest.

CIRCLE 15 ON READER SERVICE CARD

mirable simplicity. Notice how much it resembles a first generation computer performing nonoverlapped reads and writes.

Polling breaks down, however, when data from one or more sources occurs in bursts. There may be enough system capacity to handle all data presented in any given five-second interval, say, but a single device may sometimes need attention 50 times per second to avoid data overruns.

You can extend the capacity of polling systems to handle such cases in two general ways. You can build hardware that queues up data from sources that occur in bursts to tide the interface over until the control program can service it. Or you can rewrite the control program as two or more control loops, servicing devices that occur in bursts in a fast loop and performing longer computations in a slower loop.

The first approach raises hardware costs and the second raises software complexity. However, both can help you delay taking on even more hardware costs and software complexity.

A more robust solution to the problem of data that occurs in bursts is to let its hardware interface generate an interrupt whenever it needs attention. When an interrupt occurs, the computer saves whatever information is needed to restore the state of the current activity, then transfers control to a piece of software called an interrupt handler.

A properly designed handler services the attention request from the hardware interface to the point where the interface no longer feels motivated to generate an interrupt, then terminates by restoring the state of the current activity. Whatever process was running is oblivious to its interruption.

What an interrupt does, in effect, is move the synchronization point for a device further inboard from its hardware interface. You can bring the full power of the CPU to bear on buffering data to be transferred and reduce the need for special-purpose hardware in the interface. What an interrupt costs you is an added demand on the processing power of the CPU and added software complexity in dealing with the necessary synchronization.

Let's say you are using an interrupt handler to buffer data into the array *in_data*, of size *N*. You can maintain a circular buffer with an index and a count:

```
IF (N <= in_count)
   <data overrun>
ELSE
   i := in_next + in_count
   IF (N <= i)
      i := i - N
   in_data[i] := <input>
   in_count := in_count + 1
```

To process this data, you need a background loop something like:

```
FOREVER
   WHILE (in_count = 0)
      <wait>
   item := in_data[in_next]
   in_next := in_next + 1
   IF (N <= in_next)
      in_next := 0
   in_count := in_count - 1
   <process item>
```

For output devices, you write very similar code. The difference is that the background loop must *<wait>* when the output buffer is full, and the interrupt occurs when more data can be output. (Restarting an idle output device is slightly more complex, and I will gloss over the differences for now.)

But the loop just shown has a problem. Whenever an interrupt is serviced, the handler assumes that certain relationships occur among the control variables for the buffer:

- There are *in_count* items unconsumed in the buffer.
- *In_count* may be zero, but it never exceeds *N*.
- The next item to process, if present, is at *in_data[in_next]*.
- *In_next* may be zero, but it never equals or exceeds *N*.

Since interrupts may generally occur between any consecutive instructions (or even in the middle of a particularly complex instruction on some machines), the interrupt handler may be entered at any point in the background loop that is processing the input data. The handler may even be entered partway through one of the statements since high-level language statements frequently translate to several machine instructions.

You can identify any number of places where the necessary relationships do not hold among the control variables. Mucking with them at these places leads to much mayhem. The relationships hold at the beginning of the loop body. By the time execution gets to the end of the loop body, the relationships are restored.

The problem occurs in between, where the operations needed to update the state of the buffer are not atomic. Hence, this sequence of code is called a critical region. In a critical region, shared data is in an inconsistent state. Because the data is shared by agents that are proceeding

More Magic from Blaise.

Turbo C TOOLS™

Magic is easy with Turbo C TOOLS in your bag of tricks. New Turbo C TOOLS™ from Blaise Computing is a library of compiled C functions that allows you full control over the computer, the video environment, and the file system, and gives you the jump on building programs with Borland's new C compiler. Now you can concentrate on the creative parts of your programs.

The library comes with well-documented source code so that you can study, emulate, or adapt it to your specific needs. Blaise Computing's attention to detail, like the use of function prototyping, cleanly organized header files, and a comprehensive, fully-indexed manual, makes Turbo C TOOLS the choice for experienced software developers as well as newcomers to C.

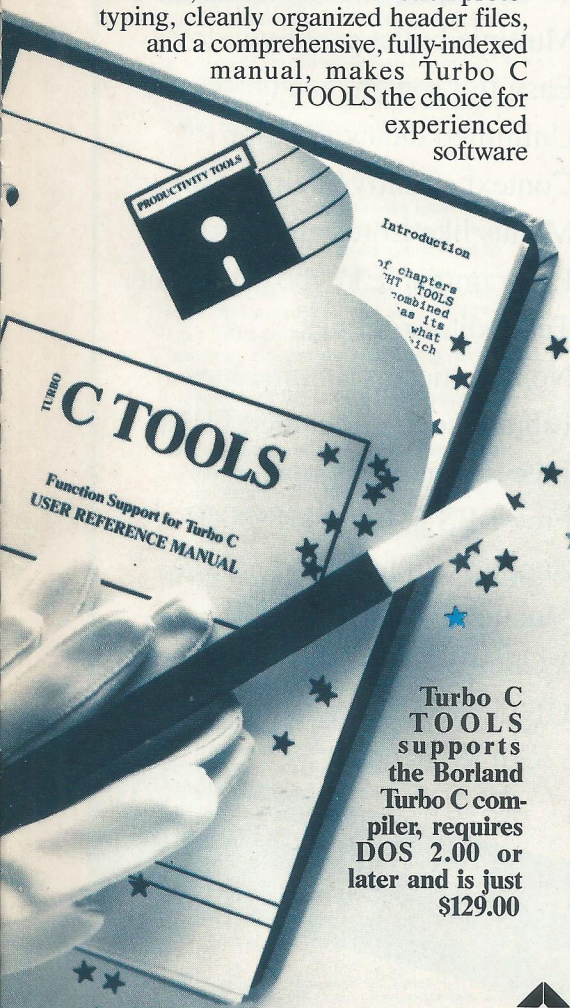
Turbo C TOOLS provides the sophisticated, bullet-proof capabilities needed in today's programming environment, including removable windows, "sidekickable" applications, and general interrupt service routines written in C.

The functions contained in Turbo C TOOLS are carefully crafted to supplement Turbo C, exploiting its strengths without duplicating its library functions. As a result you'll get functions written predominantly in C, that isolate hardware independence, and are small and easy to use.

Turbo C TOOLS embodies the full spectrum of general purpose utility functions that are critical to today's applications. Some of the features in Turbo C TOOLS are:

- **WINDOWS** that are stackable and removable, that have optional borders and a cursor memory, and that can accept user input.
- **INTERRUPT SERVICE ROUTINE** support for truly flexible, robust and polite applications. We show you how to capture DOS critical errors and keystrokes.
- **INTERVENTION CODE** lets you develop memory resident applications that can take full advantage of DOS capabilities. With simple function calls, you can schedule a Turbo C function to execute either when a "hot key" is pressed or at a specified time.
- **RESIDENT SOFTWARE SUPPORT** lets you create, detect, and remove resident utilities that you write with Turbo C TOOLS.
- **FAST DIRECT VIDEO ACCESS** for efficiency, and support for all monitors including EGA 43-line mode.
- **DIRECTORY AND FILE HANDLING** support let you take advantage of the DOS file structure, including volume labels and directory structure.

In addition to Turbo C TOOLS, Blaise Computing Inc. has a full line of support products for Microsoft, Lattice and Datalight C, Microsoft Pascal and Turbo Pascal. Call today for details, and make magic!



Turbo C TOOLS supports the Borland Turbo C compiler, requires DOS 2.00 or later and is just $129.00

THE BLAISE MENU

Turbo POWER TOOLS PLUS *$99.95*
Screen and window management including EGA support; DOS memory control; ISRs; scheduled intervention code; and much more. For Turbo Pascal.

Turbo POWER SCREEN
COMING SOON! General screen management; paint screens; block mode data entry or field-by-field control with instant screen access. For Turbo Pascal.

Turbo ASYNCH PLUS *$99.95*
Interrupt driven support for the COM ports. I/O buffers up to 64K; XON/XOFF; up to 9600 baud; modem and XMODEM control. For Turbo Pascal.

PASCAL TOOLS/TOOLS 2 *$175.00*
Expanded string and screen handling; graphics routines; memory management; general program control; DOS file support and more. For MS-Pascal.

C TOOLS PLUS *$175.00*
Windows; ISRs; screen handling; multiple monitors; EGA 43-line text mode; direct screen access; DOS file handling and more. For MS and Lattice C version 3.00 and later.

LIGHT TOOLS *$99.95*
Windows; ISRs; EGA 43-line text mode; direct screen access; DOS file handling and more. For the Datalight C compiler.

ASYNCH MANAGER *$175.00*
Full featured interrupt driven support for the COM ports. I/O buffers up to 64K; XON/XOFF; up to 9600 baud; modem control and XMODEM. For C or MS-Pascal.

VIEW MANAGER *$275.00*
General screen control; paint screens; block mode data entry or field-by-field control with instant screen access. For C or MS-Pascal.

RUNOFF *$49.95*
Text formatter for all programmers; flexible printer control; user-defined variables; index generation; general macro facility. Crafted in Turbo Pascal.

EXEC *$95.00*
NEW VERSION! Program chaining executive. Chain one program from another in different languages; specify common data areas; less than 2K of overhead.

TO ORDER CALL TOLL FREE
800-333-8087
TELEX NUMBER-338139

YES! I want to make magic!
Enclosed is $_____ for _____ copies of _____.
☐ Please send me more information on your products.
CA residents add Sales Tax. Domestic orders add $4.00 for UPS shipping, $10.00 for Federal Express standard air.
Name: _____ Phone: (_____) _____
Address: _____ State: _____ Zip: _____
City: _____ Exp. Date: _____
VISA or MC#: _____

Turbo C is a trademark of Borland International.

BLAISE COMPUTING INC.
2560 Ninth Street, Suite 316 Berkeley, CA 94710 (415) 540-5441

CIRCLE 16 ON READER SERVICE CARD

It's good for your system!

Vitamin C

"If you need source code, make sure your wallet is wide open or get VITAMIN C.

Picking the best value package is hard... If you're a source code fanatic like me, VITAMIN C is preferable."

- Computer Language, June, 1987

Fast, flexible, versatile, reliable. Vitamin C delivers the vital combination software professionals demand to produce superior applications in dramatically less time. Highly efficient, professionally crafted C code provides lightning fast displays required by today's window intensive programs.

High level functions provide maximum productivity and require little supporting code. Extended versions of these routines add flexible control over specific details when necessary. Plus, Vitamin C's versatile, open ended design is full of hooks so you can intercept and plug-in special handlers to customize or add features to most routines.

VCScreen, our screen painter / code generator speeds your development even more! Simply draw your input forms using our interactive design editor and generate perfect C source code ready to compile and link with the Vitamin C library.

- ☑ Professional C function library
- ☑ 30 day money back guarantee
- ☑ Multiple bullet proof windows
- ☑ Easy full screen data entry
- ☑ Unlimited data validation
- ☑ Context sensitive help manager
- ☑ Menus like Lotus and Mac
- ☑ Programmable keyboard handler
- ☑ Text editor routines
- ☑ No royalties or runtime fees
- ☑ Library source included FREE
- ☑ Free technical support
- ☑ Free BBS at (214)418-0059
- ☑ Supports all major compilers including Microsoft 5.0
- ☑ VCScreen code generator too!
- ☑ UNIX version available, call for details

Vitamin C $225
Includes all source code FREE! For IBM PC, XT, AT, PS/2 and true compatibles. Specify compiler when ordering.

VCScreen $99.95
For IBM PC, XT, AT, PS/2 and true compatibles. Requires Vitamin C.

We ship UPS. Please include $3 for ground, $6 for 2-day air, $20 for overnight, or $30 if outside the U.S. Texas residents add 7 1/4 % sales tax. All funds MUST be in U.S. dollars drawn on a U.S. bank. Visa & MasterCard accepted.

ORDER NOW!
(214)416-6447

creative PROGRAMMING
Box 112097 • Carrollton, Texas 75011

Windows • Data Entry • Menus • Help • Text Editing
Plus... All Source Code FREE!

asynchronously, it is never safe for more than one of the agents to be in a critical region for a given collection of data at the same time.

The interrupt handler contains a similar region of code, but it will always be executed atomically (unless there is a higher-priority interrupt that mucks with the same variables and can preempt execution). What you must do with a critical region is find a way to make it atomic.

Two fundamental approaches to making a critical region atomic are ensuring that no other concerned agents can execute while your program is executing a critical region and adding control variables that can be updated atomically to inform other agents when the critical region is busy. These agents must then elect to stay out of the critical region while it is busy.

Here, the other agent is the interrupt handler. In this case, you must dismiss the second approach because an interrupt handler has very little latitude. It must dispose of its latest input and return control, with a minimum of fuss, for two reasons:

1. The more an interrupt handler fusses around, the more likely it will cause a data overrun while it is preoccupied.

2. Any constructive fussing it might do will just involve additional synchronization problems somewhere else.

To keep interrupt handlers from interrupting critical regions, you must keep them from executing. Computers can generally disable or inhibit interrupts until a more opportune time, when the interrupts are enabled or permitted. The need to interrupt is not forgotten, its servicing is merely postponed. The background loop must be rewritten as:

```
FOREVER
   WHILE (in_count = 0)
      <wait>
   <disable interrupt>
   item := indata[in_next]
   in_next := in_next + 1
   IF (N <= in_next)
      in_next := 0
   in_count := in_count - 1
   <enable interrupt>
   <process item>
```

This code still assumes that *in_count* can be compared to zero with an operation that is either atomic or fail-safe. If you doubt this assumption, you must make the protected region even larger. Naturally, it is important to keep the amount of time interrupts are disabled and handlers are active to a minimum. Poor design or coding here can increase data overruns dramatically.

You should notice once more how much this program resembles a first generation computer performing nonoverlapped reads and writes. Most of the loop body in the previous example is just our old friend, the *READ* statement, in a new disguise.

Those of you who know something about the technology of synchronization may have noticed something odd. I have made it through most of a column without discussing gates, semaphores, queues, monitors, or any of the other higher-level concepts that fill many books and papers about synchronization. My reasons for this are:

- I wanted to pound home the historical lesson that first generation computing, isolated from synchronization issues, is the safest model for most program logic.
- I wanted to illustrate the nuts-and-bolts issues even the simplest computer interface must address.
- Most of the literature on synchronization deals with cooperating sequential processes that have more or less identical stature, whereas interrupts and background loops are intrinsically asymmetric.

There is lots more to talk about on this thorny topic. For instance, what happens when you have multiple interrupts to contend with? The best code for handling these cases is not always a simple generalization of the single interrupt case illustrated here.

Nor are your worries over once you have protected all your critical regions and made all your shared data accesses atomic. That just gets you in the game. Then you have to make sure the sequences of synchronized operations you specify make sense and never lead to deadlock.

We will consider these and other issues in the next column.

P.J. Plauger has coauthored several popular textbooks with Brian Kernighan, including The Elements of Programming Style *and* Software Tools in Pascal. *He is secretary of X3J11, the ANSI C standard committee, and convenor of the ISO C standard committee. He is also the president of Whitesmiths, Ltd.*

UNIX/C WINDOW DEVELOPMENT COMPATIBILITY with CURSES for MS-DOS and MS-OS/2.

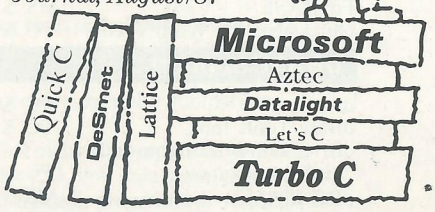
THE BETTER PRODUCT. "Aspen Scientific's Curses library is a fine PC version of System V Curses. Screen updating was fast and clean... has good documentation and is available for many compilers...less expensive... its greater flexibility makes it an attractive package for developers." *Computer Language June/87*

"This is a nice product. If you need Unix-compatible screen output in your programs, or if you just want a nice clean window-management package, I'd recommend it." *Allen Holub, Dr. Dobb's Journal, August/87*

FREE - FAST UNIX compatible forms tool kit with source code. Included if you *ORDER RIGHT NOW.*

Complete curses tool kit: $119. Source code available for: $289.

Look at a few of the applications our Curses is benefiting: Aerospace Robotics Consulting Telephone Switching AI Voice Recognition Financial Engineering Word Processing



ASPEN SCIENTIFIC
P.O. BOX 72 WHEAT RIDGE, COLORADO 80034-0072
(303) 423-8088

Trademarks: MS-DOS, MS-Windows, MS-OS/2, XENIX, Quick C (Microsoft); Unix (AT&T Bell Labs); Turbo C (Borland); Aztec (Manx); Lattice (Lattice); Let's C (Mark Williams); DeSmet (DeSmet Software); Datalight (Datalight).

CIRCLE 18 ON READER SERVICE CARD

THE PROGRAMMER'S SHOP

helps save time, money and cut frustrations. Compare, evaluate, and find products.

RECENT DISCOVERY

OS/286 & 386 - Access 16M RAM in 286, 386 native mode. DOS interface, access. Use existing compilers or 32-bit FORTRAN, C, PASCAL. PC $ 495

AI-Expert System Dev't

Arity Combination Package	PC	$ 979
System - use with C	MS	$ 229
Auto-Intelligence	PC	$ 739
CxPERT - shell for C	MS	$ 259
Experteach II	PC	$ 339
Exsys	PC	$ 289
Level 5	MS	$ 659
Intelligence/Compiler	PC	$ 739
T.I.: PC Easy	PC	$ 435
Personal Consultant Plus	PC	$2589
Turbo Expert-Startup(400 rules)	PC	$ 129
Corporate (4000 rules)	PC	$ 359

AI-Lisp

PC Scheme LISP - by TI	PC	$ 85
Star Sapphire	MS	$ 459
TransLISP - learn fast	MS	$ 79
TransLISP PLUS	MS	$ 149

Others: IQ LISP ($155), IQC LISP ($269)

AI Prolog

APT - Active Prolog Tutor - build applications interactively	PC	$ 49
ARITY Prolog - Interpreter	PC	$ 229
COMPILER/Interpreter-EXE	PC	$ 569
MicroProlog Prof. Comp./Inter.	MS	$ 439
Prolog-86 - Learn Fast	MS	$ 89
Prolog-86 Plus - Develop	MS	$ 199
TURBO PROLOG by Borland	PC	$ 69
Turbo Prolog Toolbox	PC	$ 69

Basic

BAS_C - economy	MS	$ 179
BAS_PAS - economy	MS	$ 135
Basic Development System	PC	$ 105
Basic Development Tools	PC	$ 89
BetterBASIC	PC	$ 129
Exim Toolkit - full	PC	$ 45
Finally - by Komputerwerks	PC	$ 85
Inside Track	PC	$ 49
Mach 2 by MicroHelp	PC	$ 55
Peeks 'n Pokes	PC	$ 35
QBase Screens	MS	$ 79
QuickBASIC	PC	$ 69
Quick Pak - by Crescent Software	PC	$ 59
Quick-Tools by BC Associates	PC	$ 109
True Basic	PC	$ 79
Turbo BASIC - by Borland	PC	$ 69
Turbo BASIC Database Toolbox	MS	$ 69

FEATURES

XENIX 386 Toolkit by Santa Cruz. Tools & OS kernel support 4 Gigabyte address space, demand paging, virtual memory paging. Includes MS C, MASM, debugger, file utilities, link kit, runtime library. PC $ 379

HI-SCREEN XL - unlimited overlapping windows, pulldown, popup, Lotus menus, on-line help support. Field checking, scrolling. Screen capture, more. Multi-language support. PC $ 129

Note: Mention this ad. Some prices are specials. Ask about COD and POs. Formats: 3" laptop now available, plus 200 others. UPS surface shipping add $3/item. All prices subject to change without notice.

"FREE" Software

Instant Programmer's Help for MSDOS developers by Second City Software. Memory-resident (TSR) utility lets you access valuable programmer's reference material from within any program. OS services, BIOS calls, ASCII codes, line and box drawing characters, video modes, keyboard scan codes, interrupts, 8088 assembly language, memory maps, FCB.

Normally $79. With any order over $250 by 11/30/87 get **Instant Programmer's Help** (with 2-month life span) **FREE.**

Our Services:

- Programmer's Referral List
- Compare Products
- Help find a Publisher
- Evaluation Literature FREE
- BBS-7 PM to 7 AM 617-740-2611
- Dealers Inquire
- Newsletter
- Rush Order
- Over 700 products
- National Accounts Center

Cobol

Microfocus Cobol I/Q	PC	$ 395
Cobol spII	PC	$ 329
Level II COBOL	MS	$ 279
Personal COBOL	PC	$ 119
Professional Cobol	PC	$1195
Microsoft COBOL	MS	$ 429
COBOL Tools	PC	$ 209
Realia - very fast	MS	$ 895
PCDT by Procode	MS	$ 949
Ryan McFarland COBOL	MS	Call
COBOL-85	MS	Call
Screenplay by Flexus	PC	$ 119

C Language-Compilers

AZTEC C86 - Commercial	PC	$499
C86 PLUS - by CI	MS	$359
Datalight Optimum-C	MS	$ 99
with Light Tools by Blaise	PC	$168
Lattice C - from Lattice	MS	$269
Let's C Combo Pack	PC	$ 99
Microsoft C 5.0- Codeview	MS	$275
Microsoft Quick C	MS	$ 69
Rex - C/86 - standalone ROM	MS	$695
Turbo C by Borland	PC	$ 69

C Libraries-Communications

Asynch by Blaise	PC	$ 125
Essential Comm Library	PC	$ 125
With Breakout Debugger	PC	$ 189
Greenleaf Comm Library	PC	$ 129
Multi-Comm - add multitasking	PC	$ 135

DBASE Language

Clipper compiler	PC	$ 399
dBASE II	MS	$ 329
dBase III Plus	PC	$ 429
dBASE III LANPack	PC	$ 649
DBXL Interpreter	PC	$ 139
FoxBASE+ - V2.0	MS	$ 289
Quicksilver by Word Tech	PC	$ 499

DBASE Support

dAnalyst	PC	$ 89
dBase Tools for C	PC	$ 65
dBrief with Brief - Auto-Indent, views structures	PC	Call
dBC ISAM by Lattice	MS	$ 169
Documentor - dFlow superset	MS	$ 229
Genifer by Bytel-code generator	MS	$ 279
QuickCode III Plus	MS	$ 239
R&R Report Generator	MS	$ 139
Seek-It - Query-by-example	PC	$ 79
Silver Comm Library	MS	$ 139
Tom Rettig's Library	PC	$ 79
UI Programmer-user interfaces	PC	$ 249

RECENT DISCOVERY

The Baler - compiles Lotus 1-2-3 (1A, 2.01, or VP-Planner). Protect code, speed execution, save disk space, password capability, user does not need 1-2-3. **No Royalties.** PC $ 459

Editors for Programming

BRIEF Programmer's Editor	PC	Call
EMACS by UniPress Source:	$895	$ 265
Epsilon - like EMACS	PC	$ 149
KEDIT - like XEDIT	PC	$ 99
ME Macro Editor - Source	PC	$ 79
Micro Focus Micro/SPF	PC	$ 139
PC/EDT - macros	PC	$ 229
PC/VI - by Custom Software	MS	$ 109
Personal REXX - V1.6	PC	$ 99
PMATE - power, multitask	PC	$ 109
SPF/PC - Version 2.0	PC	$ 189
Vedit	MS	$ 99
Vedit PLUS	MS	$ 129

C Language-Interpreters

C-terp by Gimpel - full K & R	MS	$219
C Trainer - by Catalytix	PC	$ 89
INSTANT C - Source debug, Edit to Run-3 seconds, .OBJs	MS	$369
Interactive C by IMPACC Assoc.	PC	$189
Run/C Professional	MS	$155
Run/C Lite	MS	$ 79

C Libraries-General

Blackstar C Function Library	PC	$119
C Essentials - 200 functions	PC	$ 75
C Tools Plus (1 & 2) - Blaise	PC	$135
C Utilities by Essential	PC	$119
Entelekon C Function Library	PC	$119
Entelekon Superfonts for C	PC	$ 45
Greenleaf Functions-portable, ASM		$139
LIGHT TOOLS by Blaise	PC	$ 69

C Libraries-Files

C Index by Trio	MS	$ 89
/File is object only	MS	$ 89
/Plus is full source	MS	$319
BTree by Soft Focus	MS	$ 69
CBTREE - Source, no royalties	MS	$ 99
CTree by Faircom - no royalties	MS	$315
ctree w/ rtree	MS	$519
rtree - report generation	PC	$239
dB2C Toolkit V2.0	MS	$249
dbQUERY - ad hoc, SQL - based	MS	Call
dbVISTA - full indexing, plus optional record types, pointers, Network.		
Object only - MS C, LAT,	C86	Call
Source - Single user	MS	Call
Source - **Multiuser**	MS	Call
dBx - translator	MS	$299
w/source to library	MS	$429

FEATURES

HUMMINGBOARD 386-Develop 2.6 or 7.9 times faster than a 8 MHZ AT. AT or XT addin board uses dual processors for Speed and Hardware Debugging. 16 MHZ or 20 MHZ. Call about Benchmarks, Trial Program.

Greenleaf C Sampler - 107 functions for communications, windows, menus, business graphics, keyboard control from popular Greenleaf libraries. Turbo Quick C support. No royalties. PC $169

We support MSDOS (not just compatibles), PCDOS, Xenix-86, CPM-80, Macintosh, Atari ST, and Amiga.

THE PROGRAMMER'S SHOP

provides complete information, advice, guarantees and every product for Microcomputer Programming.

.USE The "Right" Dbase Tools

Applications programmers - These products turn Dbase into a professional applications development environment. Generate code fast, test interpretively, debug, create reports, compile - even access from C.

Find out more about the wealth of Dbase tools that exist to enrich your environment and raise productivity. Call one of our Specialists TODAY, ask for a FREE "Dbase Programmer's Catalog."

Call before November 30, 1987 and mention this ad for these SPECIAL PRICES:

	List	Normal	SPECIAL
Dbase Prof. Dev. Lib.	$195	$179	**$159**
dBC III (with source)	$500	$389	**$299**
dBXL by WordTech	$169	$129	**$ 99**
dFLOW - analyze, debugger	$149	$125	**$109**
Foxbase + V2.0 Compiler	$395	$289	**$269**
Genifer - code generator	$395	$299	**$269**
Quicksilver - compiler	$599	$399	**$369**
R&R Relational Report Writer	$150	$139	**$119**

C Support-Systems

Advantage C++	PC	$479
C Sharp - **realtime,** tasks.	PC	$495
C ToolSet - DIFF, xref, source	MS	$ 89
Lattice Text Utilities	MS	$ 89
Multi-C - multitasking	PC	$135
PC LINT-Checker. Amiga $89	MS	$ 99
Pfantasy Pac - by Phoenix	PC	$599
PforCe++	PC	$215
Pre-C - Lint-Like	MS	$155
Quickshell - script compiler	PC	$349
Sapiens V8 - 8M workspace	PC	$269
SECURITY LIB - Source $229	PC	$115
Timeslicer	PC	$265
with library source	PC	$895

C-Screens, Windows, Graphics

C Power Windows by Entelekon	PC	$109
C Worthy Interface Library	PC	$249
Curses by Aspen Scientific	PC	$109
Curses by Lattice	PC	$ 89
dBASE Graphics for C	PC	$ 69
ESSENTIAL GRAPHICS - fast	PC	$185
FontWINDOW/PLUS	PC	$229
GraphiC - new color version	PC	$279
Greenleaf Data Windows	PC	$159
w/source	PC	$269
LightWINDOWS/C-for Datalight C	PC	$ 79
Screen Ace Form Master	PC	$195
TurboWINDOW/C - for Turbo C	PC	$ 79
Vitamin C - screen I/O	PC	$159
Windows for C - fast	PC	$189
Windows for Data - validation	PC	$319
View Manager - by Blaise	PC	$199
ZView - screen generator	MS	$129

Debuggers

386 Debug - by Phar Lap	PC	$129
Breakout - by Essential	PC	$ 89
CODESMITH - visual	PC	$ 99
C SPRITE - data structures	PC	$119
DSD87	PC	$ 89
Periscope I	PC	$289
Periscope II	PC	$139
Periscope II-X	PC	$105
Periscope III - 10 MHZ Version	PC	$899
Pfix-86 Plus - by Phoenix	PC	$225
Turbo TDebug	PC	$ 49
SoftProbe II - embedded systems	PC	$695

Translator

dB2C - requires toolkit	MS	$249
RTC PLUS by Cobalt Blue	MS	$399
TP2C	PC	$219
Turbo-to-C-Tools by TGL	PC	$109

Fortran & Supporting

50:More FORTRAN	PC	$ 95
ACS Time Series	MS	$389
I/O Pro - screen development	PC	$129
MACFortran by Microsoft	MAC	$229
MS Fortran - 4.0, full 77'	MS	$279
No Limit - Fortran Scientific	PC	$109
Numerical Analyst by Magus	PC	$249
PC-Fortran Tools - xref, pprint	PC	$165
RM/Fortran	MS	Call
Scientific Subroutines - Matrix	MS	$129

Multilanguage Support

BTRIEVE ISAM	MS	$185
BTRIEVE/N-multiuser	MS	$455
Flash-Up Windows	PC	$ 79
GSS Graphics Dev't Toolkit	PC	$375
Hoops Graphics Library	PC	$549
Informix 4GL-application builder	PC	$789
Informix SQL - ANSI standard	PC	$639
NET-TOOLS - NET-BIOS	PC	$129
Opt Tech Sort - sort, merge	MS	$ 99
PANEL	MS	$215
Panel Plus	MS	$395
Pfinish - by Phoenix	MS	$229
PolyBoost - speed I/O, keyboard	PC	Call
Polylibrarian by Polytron	MS	Call
PVCS Corporate or Personal	MS	Call
Report Option - for Xtrieve	MS	$109
Screen Sculptor	PC	$ 89
SSP/PC - 145+ math routines	PC	$269
Synergy - create user interfaces	MS	$375
Xtrieve - organize database	MS	$199
ZAP Communications - VT 100	PC	$ 89

Pascal and Supporting

ALICE - learn Pascal	PC	$ **49**
Exec - Chain Programs	MS	$ 79
Marshall Pascal	MS	$155
MetaWINDOWS-graphics tools	PC	$159
MetaWINDOWS PLUS	PC	$229
Microsoft PASCAL - faster	MS	$189
Pascal 2 - by Oregon Software	MS	$319
System Builder	PC	$129
Turbo Extender by Turbo Power	PC	$ 65
TurboHALO	PC	$ 95
Turbo Optimizer - object only	PC	$ 59
Turbo Pascal	MS	$ 65

DataBase & File Management

DataFlex by Data Access	PC	$ 899
Dataflex multiuser	PC	$1149
Magic PC	PC	$ 179
Paradox	PC	$ 549
Revelation by Cosmos	PC	$ 779

RECENT DISCOVERY

ACTOR by Whitewater Group - Graphic, object-oriented language for prototyping, interactive programs and MS Windows applications. Fast feedback, integrated editor, debugger. Call to multiple languages. PC $ 419

Other Languages

APL*PLUS/PC	**PC**	**$ 369**
CCS Mumps - Multiuser	PC	$ 359
CCS Mumps - Single user	PC	$ 49
Microsoft MASM	MS	$ 98
Modula-2 - V3.0 Compiler Pack	PC	$ 89
Modula-2 - V3.0 Dev. System	PC	$ 219
PC Forth + - by Lab Micro	PC	$ 189
Smalltalk/V	MS	$ 85
SNOBOL4 + - great for strings	MS	$ 80
UR/Forth	MS	$ 259

Xenix/Unix

Basic - by Microsoft		$ 209
C-Terp by Gimpel Software		$ 379
Cobol - by Microsoft		$ 609
Cobol Tools - by Microsoft		$ 319
dBQuery - multi source		$ 849
Fortran or Pascal - by Microsoft		$ 419
Foxbase +		$ 649
MicroFocus Lev. II Compact COBOL		$ 795
Panel		$ 535
RM/Cobol or RM/Fortran		Call
Vedit +		$ 209
Xenix Complete System		$ 989
Xenix Development System		$ 489

Other Products

386 Assembler/Linker	PC	$ 389
386/DOS Extender	PC	$ 919
Advantage Link	PC	$ 359
ASMLIB - 170+ routines	PC	$ 125
asmTREE - B + tree file mgmt.	PC	$ 329
Back-It by Gazelle	MS	$ 119
Beyond.Bat - loop, branch	PC	$ 85
Dan Bri klin's Demo Program	PC	$ 59
Desqview by Quarterdeck	PC	$ 109
Disk Technician - smart upkeep	PC	$ 89
Fast Back-V1.5 for PS/2	PC	$ 129
Help/Control - on line help	PC	$ 99
Interactive Easyflow V5.0	PC	$ 125
Link & Locate - Intel tools	MS	$ 329
LMK - like UNIX make	MS	$ 139
Microsoft Windows	PC	$ 69
Software Development Kit	PC	$ 319
MKS Toolkit - Unix, vi, awk	PC	$ 115
Norton Commander	PC	$ 55
Norton Utilities Advanced	MS	$ 99
PLink - 86 PLUS - overlays	MS	$ 279
Polymake by Polytron	MS	Call
PolyShell by Polytron	MS	Call
Q-Dos II by Gazelle	PC	$ 59
RPG II Compiler	PC	$ 639
Sapiens V8 - 8M virtual mgr.	PC	$ 269
Source Print - V 3.0	PC	$ 75
Taskview - ten tasks	PC	$ 55
Tree Diagrammer	PC	$ 65
Visible Computer: 80286	PC	$ 89
Xtree - classic graphic tree	PC	$ 45

Call for a catalog, literature, advice and service you can trust

VISA HOURS MasterCard

8:30 AM - 8:00 PM EST.

800-421-8006

THE PROGRAMMER'S SHOP ™

Your complete source for software, services and answers

5-L Pond Park Road, Hingham, MA 02043

Mass: 800-442-8070 or 617-740-2510 9/87

"I would like to mention that I appreciate the way that The Programmer's Shop does business. It is indeed refreshing to be able to call and get answers that you can trust in, to questions on various products."

Donald E. Winters
MIS Software Development Inc.

CIRCLE 19 ON READER SERVICE CARD

We've listed 10 Important Reasons C Programmers use Our db_VISTA™ File Manager, and now there's one more!

C Programmer's Power Toolkit

Now build fast, portable, royalty-free applications easily!

db_VISTA™ File Manager 2.21 &

This is the file manager that has out-performed other data management systems in its class, yet provides capabilities far beyond the rest! Faster than the competition, db__VISTA—the File Manager for Professional C programmers, is also one of the most powerful systems for data organization and access available. You'll like the design flexibility and the speed! db__VISTA is written in C for maximum portability.

C-scape 2.0 with Look&Feel™

the WYSIWYG Screen Design Tool

This is the state-of-the-art interface management system preferred by professional C programmers and consultants worldwide. Look & Feel makes design easy because you draw right on the screen! Then it converts those designs to C-scape calls. No more complicated coding! Create menus, data entry screens, pop-up windows, help screens and more.

WRITTEN IN C • PORTABLE • SOURCE CODE AVAILABLE
ROYALTY FREE • EXTENSIVE FUNCTIONS • EASY TO LEARN & USE • FLEXIBLE
FREE TECH SUPPORT • 30 DAY MONEY BACK GUARANTEE • AND MORE!

db_VISTA™ File Manager 2.21

Look at these db_VISTA™ features:

- Fast Access to data with minimum overhead
  - B-tree indexing
  - network database model
- Transaction processing
- File locking
- Single/Multi-user LAN options
- Data Definition Language patterned after C
- SQL-based db__QUERY available
- File transfer utilities for ASCII & dBASE files
- Interactive database utility
- Upward database compatibility

C-scape 2.0 with Look&Feel™

Look at these Look&Feel™ features:

- Generates readable C code
- Create menus and data entry screens
- Define fields of any type
- Variables, prompts, and validation
- Line draw and erase
- Block, move, cut, paste, copy
- Horizontal and vertical scrolling
- Edit Dan Bricklin Demo slides
- Full color support
- Fast and easy to use
- Includes help
- Full feature demo available

Look at these C-scape 2.0 features:

- Windows, menus, smart borders
- Vast help system
- Create any type of field
- Data entry and validation
- Extensive function library
- Swappable device drivers
- Easy to learn & use
- Easy to maintain and modify
- Complete 500 page manual

Environments:

Both db__VISTA and C-scape support popular Operating Systems like MS-DOS & UNIX. C Compilers include Microsoft, Lattice, Turbo, Aztec and others. Call for information on other OS.

Technical Support: FREE!

We provide technical support in the form of telephone hotline service, 24 hr. bulletin board and we'll even assist with applications! (Extended services available include application development, consulting, training and support. db__VISTA offers free technical support for 60 days only.)

C PROGRAMMER'S POWER TOOLKIT	**only $425**
db__VISTA File Manager	starts at $195
C-scape with Look & Feel	$279

(For further database capabilities, inquire about db__VISTA III, the Database Development System that C Programmer's designed!)

To Order:

Call us today; we'll answer any questions and get you started!

Call Toll-Free Today!

MasterCard **1-800-db - RAIMA** VISA

(that's 1-800-327-2462)

RAIMA™ CORPORATION

3055 112th Avenue N.E., Bellevue, WA 98004 (206) 828-4636
Telex: 6503018237MCIUW FAX: (206) 828-3131

CL 11/87

DESIGNING WITH DATA BASES

Screen and report generators

By Matthew Rapaport

In the last issue, I began to cover a variety of tools associated with DBMSs called nonprocedural languages. I focused on list processors such as the *LIST* commands in Cosmos's Revelation and Dunn and Bradstreet Corp.'s NOMAD2, and I discussed the output capabilities of IBM Corp.'s Standard Query Language (SQL). I noted that SQL is capable not only of output from the data base, but input and update operations on the data as well.

The SQLs of Informix Software Corp.'s Informix and Unify Corp.'s UNIFY have this capability. In addition, Informix's SQL is capable of creating new data structures or modifying old ones, which makes it the only query language that meets all of E.F. Codd's specifications for the functional capability of a relational DBMS's nonprocedural interface.

The trade-offs for the flexibility of a full-featured SQL (as compared to a pure list processor) are a relatively complex syntax and the inability to create complex output formats (like checks replete with names, amounts, stubs, etc.) and perform complex input and update transactions involving data manipulation on a record-by-record basis. Another weakness of input with SQLs is that the operator is required to know precisely the order of fields in the external view or base relation. Input to a relation using SQL takes the following form:

```
INSERT EMPLOYEE JONES, JOHN,
    05/01/86, SENIOR MANAGER,
    48000.00, 10/
```

This input creates a new record in the relation *EMPLOYEE* with fields *LAST_NAME, FIRST_NAME, HIRE_DATE, TITLE, SALARY,* and *DEPARTMENT,* respectively. If the operator had inadvertently reversed the *TITLE* and *SALARY* fields, the insert would at best fail (if the DBMS was doing proper integrity checks) and at worst cause instant corruption of the data base.

Input in this manner is very useful during the development and testing of a new data structure and markedly enhances where preformatted data is fed to a batch SQL script. SQL is cumbersome and prone to error if used in on-line production applications.

This type of input may also be entirely untenable in large data bases (a typical production employee data base may have 200 or more fields describing each employee). For this reason, and because production applications frequently require reports with elaborate titles, footings, or other page formats, two other specialized nonprocedural languages and/or utilities have become standard accoutrements of most DBMSs: screen and report generators.

Of the these two tools, screen generators are generally easier to use. Screens can be formatted randomly. They require titles and appropriate prompts, but otherwise data fields can be displayed or input in any order along with defined fields and other variables.

In contrast, report generators must account for the fact that output generally includes headings, footings, subtotal breaks, or other special fields that should be properly placed on hard copy that usually moves in one direction only. Screens may have to test and format data in complex ways, but this happens behind the scenes, while the results of complex manipulations must be revealed, sometimes in complicated images on paper by report generators.

Thus, while screen generators must sometimes handle intricate inserts and updates of numerous relations, report generators must not only report from one or more base tables, but also be able to do elaborate page formatting in a less forgiving medium than the CRT.

In covering a range of screen generators, it seems appropriate to start with that of Ansa Corp.'s Paradox. In some ways the Paradox user interface is a lesson in clean screen generation. To begin with, you can insert or update values in a data base through operations on data displayed in tabular format, much like that of a spreadsheet. One or more tables can be displayed on the screen at the same time. Fields that will not fit in a column can be expanded with the touch of a key, filling the screen both horizontally and vertically for extensive text entry, then instantly collapsed for entry of other fields.

A single keystroke allows scrolling up or down through multiple tables, and all data entry is instantly checked against edit criterion established when the table structure was created. Unfortunately, so many end-user application specifications demand the more traditional, one-record-at-a-time form of data entry, update, and query. Using the Paradox table format quickly becomes natural and can be a tremendous time-saver since multiple records can be edited with commits to the data base executed over multiple changes, saving disk I/O.

With Paradox, it is possible to build the traditional end-user data entry interface. The paradox forms facility allows easy creation of data entry screens, including easy paging through multiple screens of highlighted boxes and calculated values (based on a predicate relation between one or more fields in the current record, constants, or dates).

One advantage to forms is the multiple display of fields on one screen, from one record at a time, compared to a tabular layout. Paradox forms can display data from one table at a time only. Calculated fields can be useful and cannot be displayed using the tabular representation. Calculated fields such as *DUE-DATE: ([INVOICE-DATE]+30* and *OVER-DUE-CHARGE:([INVOICE-TOTAL]*[CUSTOMER-INCREASE-RATE])+[INVOICE-TOTAL]* can be established for display purposes only. They are not stored in the data base.

However, screen forms can be printed and serve well as purchase orders, invoices, or even checks, with the limitation that calculations cannot be based on reading values from other tables (for example, you cannot calculate the net amount on a paycheck by reading a tax table and deriving tax amounts).

Input, update, and deletes are single-keystroke operations, with the effective keys consistently established in the Paradox interface. As usual, security and convenience are at odds with one another.

One of the problems with Paradox's easy access to data is that *DELETE* operations cannot be shut off unless one goes to the trouble of programming the screen directly via Paradox's PAL language. While PAL is a good language (and will be discussed in a future column), having to use it defeats the whole purpose of having such a powerful nonprocedural screen interface to begin with.

Paradox illustrates how a given product, through the interface philosophy its creators choose to implement, is best suited to one niche only, even though its underlying DBMS is capable of handling the data structures and manipulations of dissimilar environments. For example, Paradox is easily capable of handling the payroll requirements of a small- or medium-sized company, but the sensitivity and historical significance of the data prevents data-entry operators from using the facilities of the entire Paradox nonprocedural user interface.

In contrast, an application with a strong management information systems (MIS) leaning is ideally suited to the Paradox philosophy. Here data may be deleted and manipulated in easily created temporary tables for "what-if" analysis or data summary.

* Before turning to the screen facilities of the next DBMS on our list, it is worth noting briefly the facilities of Borland International's Reflex in connection with spreadsheetlike screen interfaces. Reflex is not a DBMS because it cannot operate on more than one table at a time; it possesses no facilities for intertable relational edits. Like Paradox, it can display a table for entry, update, or inquiry of data. Standard entry forms can be created with one or more pages.

In many ways Reflex resembles the Paradox interface, but it has more features for single-table manipulation. For example, it instantly cross-tabulates two different fields (such as salespeople and their sales figures by month), displaying one field along the *X* axis of a table and the other on the *Y* axis. Reflex also takes multiple fields from a single table and instantly graphs their relationship in one of several ways: bar, line, pie, etc.

In addition to this, Reflex has a spreadsheetlike capability to replicate data down rows, producing new data according to a formula. For example, a given table may have rows that represent months and columns that represent sales figures for different regions. These rows can be duplicated, in effect projecting the data into the future, with sales made to vary in relation to a user-created function. In short, Reflex would be an excellent example of a versatile DBMS front end if its various capacities could automatically access the files of another vendor's underlying (and running) DBMS.

Given the ability of most DBMS systems to dump subsets of tables to ASCII files, Reflex can be used in conjunction with other systems to produce very sophisticated analytical systems. (Reflex can read ASCII files into its file system as well as several other formats directly.)

Screen Generators in other DBMS products are typically more difficult to master, but most are more powerful since they can update several tables in the same screen (or multiscreen set). Microrim's R:base System V screen generator, called Forms Express, can update five tables in one screen set. Like Paradox, R:base screens are generated with built-in code for handling insert, update, and delete activities in a single-keystroke manner.

R:base screens can be divided into regions of tiers within regions, allowing the user to display data from different tables in a combination of custom form and tabular display in the same screen. Furthermore, R:base System V provides mechanisms for restricting operations in any given screen. For example, it can prevent record deletions or allow updates but no inserts, or vice versa. It also enables automatic updating of one relation: a function is performed over an operator entry in another relation. Operator entry of a customer amount-paid in a table called CUSTOMER-RECEIPTS can be converted to bonus points, which are used to update a totaling bucket in the table CUSTOMER-BONUS-TOTALS.

The UNIX versions of Informix's and UNIFY's screen generators are not as powerful as those I've discussed. I have not yet tried the DOS versions of these products. Both will produce multipaged screens. Informix's screen generator updates multiple relations in the same screen set, but not in one screen. It cannot automatically update one relation based on an entry made to another. UNIFY's screen operator cannot handle more than one table in a screen form.

Informix screens can be customized after they are produced by the screen generator. The generator's output is an editable listing in a documented nonprocedural language. One can modify this code to move fields around the screen or between screens and to generate calculated fields that, like their Paradox counterparts, are not stored in the data base. Listing 1 is an example of the output of Informix's screen generator after customization.

The Informix screen language is capable of assigning default values, masks, enumerated types or ranges, and insert, update, and delete limits imposed on a field-by-field level. It can also assign relational edits otherwise not available in the Informix data dictionary. UNIFY's generator does not produce editable code but can create more elegant forms than the straightforward output of Informix. UNIFY screens, however, have the same limited I/O capabilities as Paradox screens; therefore a lack of ability to enhance them is not an asset.

Even with Informix's enhancement potential, such work requires knowledge of a programming language (albeit a high-level nonprocedural one). The Informix screen language is a good example of a specialized nonprocedural system. In some ways, working with it is like using the @ SAY logic of Ashton Tate's dBASE line, except that the record retrieval and update logic is built into the process by the initial screen generation and automatically connected to the appropriate keys.

The logic-handling cursor movement among the fields of the screen is also similar to that of Ashton Tate's dBASE line. When a developer customizes the code generated by Informix, *get, update,* or *delete* statements are not present.

In contrast, R:base System V has no code to customize. Its nonprocedural interface is two-dimensional, expecting the user to paint a screen with elaborate sequences of keystrokes and answers to questions posed by the utility software. Expertly working such a piece of software is just as difficult as mastering the older string language products (like Informix). Enhancing a screen in R:base means exercising the features of a substantial and dedicated editing system fully capable of turning any screen in a system into any other screen and as complex as a small-scale word processor or good line editor.

The screen generators of d:BASE III Plus and Revelation produce procedural code that can then be enhanced. Initially, these systems produce screens with simple functions, but a developer who knows how to subsequently handle their procedural languages can manipulate them. Revelation screens operate in interpreted or compiled mode.

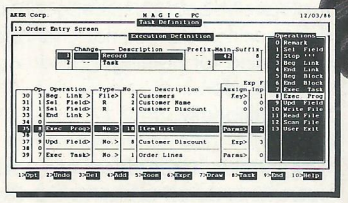
In compiled operation, the screen runs from its generated procedural code, which is compiled to the Revelation runtime system (the compiler does not produce .EXE or .COM files). In interpreted mode, Revelation screens are similar to the screens of R:base, Paradox, and UNIFY. Achieving R:base-like performance with Revelation requires somewhat more work and learning. While the

Make $500/hr.

Now develop DBMS applications 10 times faster for only $199 with MAGIC PC — or your money back!

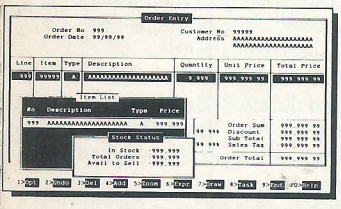
Database programmers, why waste your time hacking out code?
Imagine how much faster and more profitable you'd be if you could whip up powerful database applications without the time-consuming coding pains...
Introducing Magic PC from Aker, your professional dream come true. It's not another line-by-line syntax treadmill like any DBMS or 4GL.
Finally you can program as quickly as you design, while you delegate all the mundane and redundant coding tasks to Magic PC.

Program 10 times faster



Develop relational database applications 10 times faster using a visual design-driven interface. Instead of writing mountains of "how to" procedural code, you quickly place your program design specs in Execution Tables and Magic PC's engine executes them automatically. Don't lose any more time editing and debugging programs by hand.

Incredible Zoom power



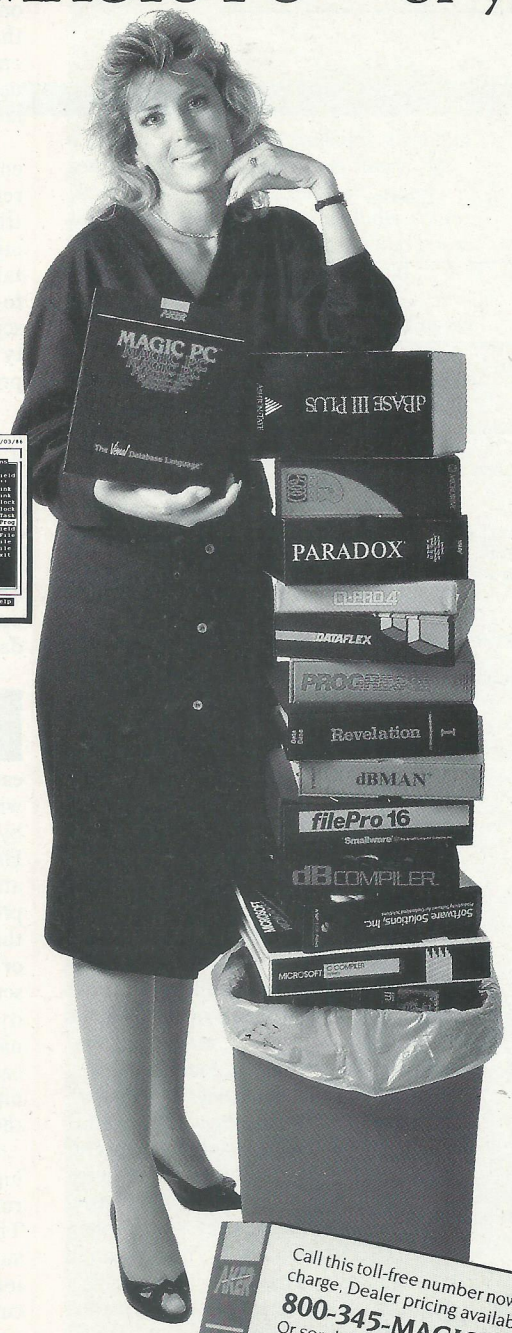
Magic PC's phenomenal Zoom power magically co-executes related programs through nested Zoom windows smoothly with auto data scrolling in all directions. While Zooming, query and transfer data across windows or even Zoom deeper.

No more maintenance!

Change your programs on the fly without any manual maintenance responsibility. Magic PC automatically updates your changes online since all the data describing your design (data dictionary, programs and menus) make up a single file, self-maintaining Integrated Library.

Magic PC does it all

Design your entire database application with only one comprehensive development system. Generate both online programs (screens, windows, menus), as well as batch programs (reports, updates, import/export, etc.) with full color and graphics. You no longer fall between the cracks dealing with separate and inconsistent programming utilities.



Free LAN features

Develop multi-user applications for local area networks with Magic PC's automatic support for file and record locking security.

Quick prototyping

Prototype a complete working application in just hours and get immediate customer feedback to finalize the design. It's a true time-saver.

Stand-alone runtime

Distribute your applications and protect your design with a low cost runtime engine. It has the friendliest end-user visual interface you've ever seen with built-in, menu-driven and syntax-free data retrieval power.

Jeff Duntemann, PC Tech Journal:

"*Magic PC is probably the best integrated database application generator that we have seen... very smooth system, and smoothness comes at a premium these days.*" Also recommended by PC Magazine, PC World, PC Week, Computer Language, Data Based Advisor and many more around the world.

Try it for $19.95

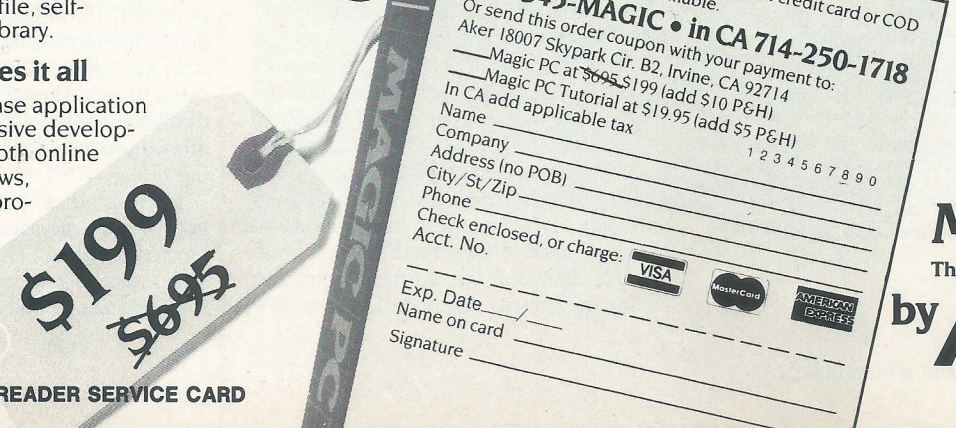
If you develop database applications for a living, you can't afford not to try Magic PC for yourself right now. For $19.95 you'll get the Magic PC Tutorial software and documentation for hands-on evaluation, complete with a step-by-step guide to develop an Order Entry sample application in just a few hours.

Magic PC ~~$695~~ $199

No kidding! For a limited time only, save almost $500 off the $695 list price, and get the complete unprotected Magic PC software for only $199 at our special introductory non-resale price.

Money back guarantee

Even at $199 you can't go wrong with our no-risk guarantee: keep it only if it makes magic for you, or we'll buy it back within 30 days less $19.95 restocking fee.



Call this toll-free number now with your credit card or COD charge. Dealer pricing available.

800-345-MAGIC • in CA 714-250-1718

Or send this order coupon with your payment to:
Aker 18007 Skypark Cir. B2, Irvine, CA 92714

___Magic PC at ~~$695~~ $199 (add $10 P&H)
___Magic PC Tutorial at $19.95 (add $5 P&H)
In CA add applicable tax 1 2 3 4 5 6 7 8 9 0
Name _____
Company _____
Address (no POB) _____
City/St/Zip _____
Phone _____
Check enclosed, or charge: VISA MasterCard AMERICAN EXPRESS
Acct. No. _____
Exp. Date ___/___
Name on card _____
Signature _____

System Requirements: IBM PC, XT, AT, PS/2 and 100% compatible. PC-DOS 2.0 or later, 512K, hard disk. All trademarks acknowledged.

MAGIC PC
The Visual Database Language
by AKER

CIRCLE 21 ON READER SERVICE CARD

```
Output of the Informix screen generator. The characters "o," "i,"
and "s" in the brackets represent the first characters of the
table names "orders," "items," and "stock." Their meaning is
derived from assignment statements following the screen layout.
Note the absence of an explicit loop, which compares to the high-
level procedural code generated by DataFlex in Listing 2.

data base stores

screen
{
--------------------------------------------------------------------------------
CUSTOMER INFORMATION

Number:     [c1          ]
Company:    [c4                    ]
First Name: [c2           ] Last Name: [c3             ]

Address:    [c5                     ]
            [c6                     ]
City:       [c7                ] State: [  ] Zipcode: [         ]
Telephone:  [               ]

--------------------------------------------------------------------------------
}
screen
{
================================================================================
Customer Number: [c1          ]    Company: [c4                    ]

ORDER INFORMATION
    Order Number: [o11           ] Order Date:  [o12        ]

    Stock Number: [i13       ]   Manufacturer:  [i16  ]
     Description: [s14           ]              [m17              ]
            Unit: [s16           ]
                                      Quantity:  [i18          ]
                                    Unit Price:  [s15               ]
                                   Total Price:  [i19               ]

SHIPPING INFORMATION
  Customer P. O.: [o20         ]   Ship Charge:  [d1       ]

         Backlog: [a]       Total Order Amount:  [d2       ]
       Ship Date: [o21       ]
       Date Paid: [o22       ]
    Instructions: [23                                              ]
}
end

tables

customer
items
stock
orders
manufact

attributes
c1 = *customer.customer_num
   = orders.customer_num;
c2 = fname,
     comments = "Please enter initial if available";
c3 = lname;
c4 = company, reverse;
c5 = address1;
c6 = address2;
c7 = city;
c8 = state, upshift, autonext,
     include = ("CA","OR","WA","NV","AZ"),
     default = "CA";
c9 = zipcode, autonext;
c10 = phone, picture = "###-###-####";
o11 = *orders.order.num = items.order.num;
o12 = order_date, default = today, format = "mm/dd/yyyy";
i13 = items.stock.num;
    = *stock.stock.num, noentry, noupdate, queryclear;
i16 = items.manu.code, lookup m17 = manufact.manu.name joining
      *manufact.manu.code, upshift, autonext;
      = *stock.manu.code, noentry, noupdate, upshift, autonext,
     queryclear;
s14 = stock.description, noentry, noupdate;
s16 = stock.unit_descr, noentry, noupdate;
s15 = stock.unit_price, noentry, noupdate;
```

Listing 1. *(Continued on page 30)*

Revelation screen generator is harder to master with than that of R:base, Revelation's can be substantially more powerful without requiring the developer to resort to code.

Although Revelation screens are limited to operations on one table at a time, this limitation is easily bypassed without resorting to code. Each field represented on the screen (including calculated fields that may be the result of elaborate operations performed by functions stored in the data dictionary) can be customized in 26 ways.

Customizations include default values, enumerations, ranges, masks, automatic referential integrity checking, and the triggering of procedural subroutines and other screens that may address different tables. This last capacity applies not only to each field in the screen but to the screen as a whole, and includes the ability to signal the triggering of subprocedures:

If a specific field has changed...
If a specific field has not changed...
If anything (or nothing) has changed on the screen...
Unconditionally....

Triggered subprocedures can perform operations on all tables in the Revelation data base.

Data Access Corp.'s DataFlex is one of the easier screen generators to get started with and can be very powerful when used by a competent DataFlex programmer. Using a word processor, the designer creates a screen of a series of literal prompts and underscores that represent the windows where data will be displayed or entered to the data base. Once the screen template has been created, a utility program prompts the operator to name each attribute to be displayed in each position (up to 10 tables can be manipulated opened concurrently, but not in the initial creation of a screen).

The output of this utility program is a high-level procedural code compiled and run by the DataFlex run-time system. This code contains built-in functions such as cursor handling and data editing, insert, update, and delete, all of which can be selectively disabled by the programmer. DataFlex's screen generator generates entry code for one table at a time and creates the data table at the same time as the screen. This means a developer cannot use the screen generator to produce a new screen for an existing table.

However, these unfortunate by-products of the generation process can easily be circumvented. The screen and its associated code can be modified with most word processors and editors, and the out-

Clarify and document your source listing and get an "organization chart" of your program's structure

with two NEW utilities from Aldebaran Laboratories, for C, BASIC, Pascal, dBASE® FORTRAN and Modula-2 programmers.

"Occasionally, a utility comes along that makes a programmer's life much easier. SOURCE PRINT is such a program. It contributes to the programmer's job by organizing code into a legible format and by helping to organize the documentation and debugging process."

— PC Magazine Sept. 16, 1986

Source Print and Tree Diagrammer both have easy-to-use menus with point-and-shoot file selection, and let you search for files containing a given string. For IBM PC and compatibles with 256K.

Join thousands of programmers who are working more efficiently using Source Print and Tree Diagrammer. Order these indispensable tools today. We ship immediately, and there's no risk with our 60-day money-back guarantee. **Order both and save. Only $155.00.**

800-257-5773 Dept. 49
In California:
800-257-5774 Dept. 49

MasterCard, VISA, American Express, COD. Add $5 for shipping/handling.

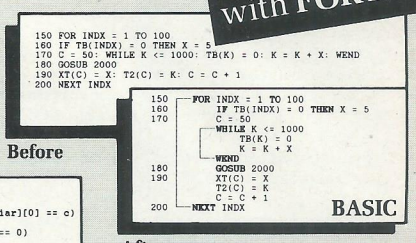
or see your local dealer!

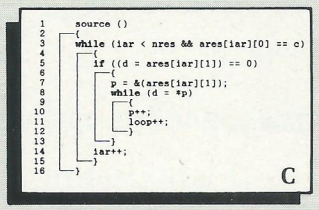
Source Print and Tree Diagrammer are trademarks of Aldebaran Labs. dBASE is a trademark of Ashton Tate. Prices subject to change without notice.

Source Print™

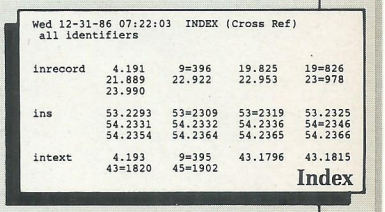
Now works with FORTRAN

organizes your source code, simplifies debugging, and makes documentation a snap! It lists one or more source files with informative page headings and optional line numbers, while offering invaluable features:





The Index (Cross-Reference list) saves you time by showing exactly where variables are used and where functions, procedures, and routines are called.



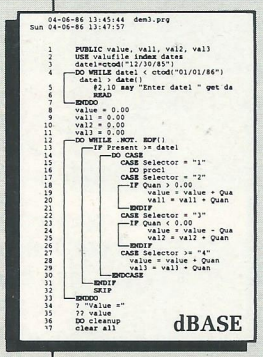
$97.00

Locations where new values may be assigned to variables are shown, making it easy to track down that mysterious value change.

Structure Outlining solves the problem of hard-to-see nested control structures by automatically drawing lines around them.

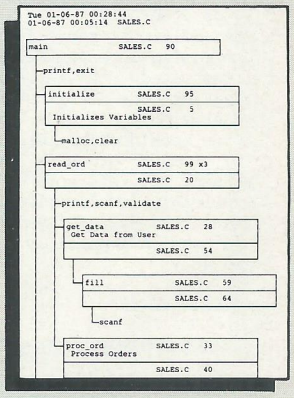
Automatic Indentation of source code and listings reduces your editing time and ensures indentation accuracy.

Plus . . . Source Print generates a table of contents listing functions and procedures. Keywords can be printed in boldface on most printers. Multi-statement BASIC lines can be split for readability. Functions and procedures can be drawn by name from one or more source files to form a new file.



Tree Diagrammer™

shows your program's overall organization at a glance. Ordinary program listings merely display functions, procedures, and subroutines sequentially, but do not display the relationships between these routines. Our revolutionary new Tree Diagrammer automatically creates an "organization chart" of your program showing the hierarchy of calls to functions, procedures, and subroutines. Recursive calls are indicated and designated comments in the source code will appear on the chart.



Tree Diagrammer helps you organize your program more logically. And you'll be amazed at how easy it is to debug when you see how your routines interact.

$77.00

Aldebaran Laboratories 3339 Vincent Rd. Pleasant Hill, CA 94523 415-930-8966

YES! Rush me ☐ Source Print @ $97. _____ ☐ Tree Diagrammer @ $77. _____
☐ Both $155. Ship/Handling $5. For CA add 6% tax _____ Total _____
Name _____
Company _____
Address _____
City _____ State _____ Zip _____
☐ Check enclosed ☐ VISA ☐ MasterCard ☐ American Express
Card # _____ Exp. Date _____
Signature _____ Phone # _____

CIRCLE 22 ON READER SERVICE CARD

```
i18 = items.quantity, include = (1 through 50),
      comments = "Acceptable values are 1 through 50";
i19 = items.total_price;
o20 = po_num, required,
      comments = "If no P.O. No. enter name of caller";
a = backlog, autonext;
o21 = ship_date, default = today, format = "mm/dd/yyyy";
o22 = paid_date, format = "mm/dd/yyyy";
o23 = ship_instruct;
d1 = displayonly type money;
d2 = displayonly type money;

instructions

customer master of orders;
orders master of items;
composites <items.stock_num, items.manu_code>
           *<stock_stock_num, stock.manu_code>

before editadd editupdate of orders
   nextfield = o20

before editadd editupdate of items
   nextfield = i13

after editadd editupdate of quantity
   let i19 = i18 * s15
   nextfield = o11

after add update query of items
  if (total of i19) <= 100 then
    let d1 = 7.50
    else
    let d1 = (total of i19) * .04

  let d2 = (total of i19) + d1

after display of orders
   let d1 = 0
   let d2 = 0

end
```

Listing 1. *(Continued from page 28)*

```
Output of the DataFlex screen generator with customizations.  The
designed screen is entered using any text editor as a template
between the "/SENSITIVE_INFO" and "/*."  Since the main file
addressed is "CUSTOMERS", the generator produces the "OPEN"
statement on line 1 and the "ENTRY" statements between lines 6
and 16.  The rest of the code is added by the programmer,
including all of the special action requirements shown in "[]" in
lines 7 thru 17.
The additional macro facilities made available for special
processing on save, delete, edit, clear, etc. are also
illustrated.  "ENTER.SAVE" is fleshed out a bit with lower-level
DataFlex code to illustrate the capacity to mix third and fourth
generation constructs, as is the "ACCEPT" statement on line 17,
which goes nowhere but is used to direct the activity of the
"ENTER.SAVE" group on lines 19 thru 30.  The line numbers are
present for reference only.  Note that there is no code
explicitly for functions like "clear," "save," and "delete."
These keys are assigned by the developer when installing DataFlex
and handled implicitly by the ENTER/ENTEREND macro loop (lines 6
and 48).

/SENSITIVE_INFO

        ORDER DATE: __/__/__

   Customer Number: _________       Customer Name
                                          First: ____________________
                                           M.I.: _
                                           Last: ____________________


   Outstanding Credit Limit: _______.__
   Total Business Last Six Months: _________.__
   Total Payments Last Six Months: _________.__
   Current Month Order Total: _______.__

   ORDER O.K.? _  ____________________________________________
/* // End of Screen Template
```

Listing 2. *(Continued on page 32)*

put of multiple runs can be concatenated in one multiscreen file set with the same editing tools and then compiled.

DataFlex's output is a collection of macro statements expanded into a run-time script code, then pseudocompiled with a fast run-time interpreter. The DataFlex macro language constitutes a fourth generation procedural language in its own right. These statements can be duplicated in a lower-level procedural language supported by the same compiler, giving the programmer more control responsibility. Developers can extend the macro language with constructs built in the lower-level syntax.

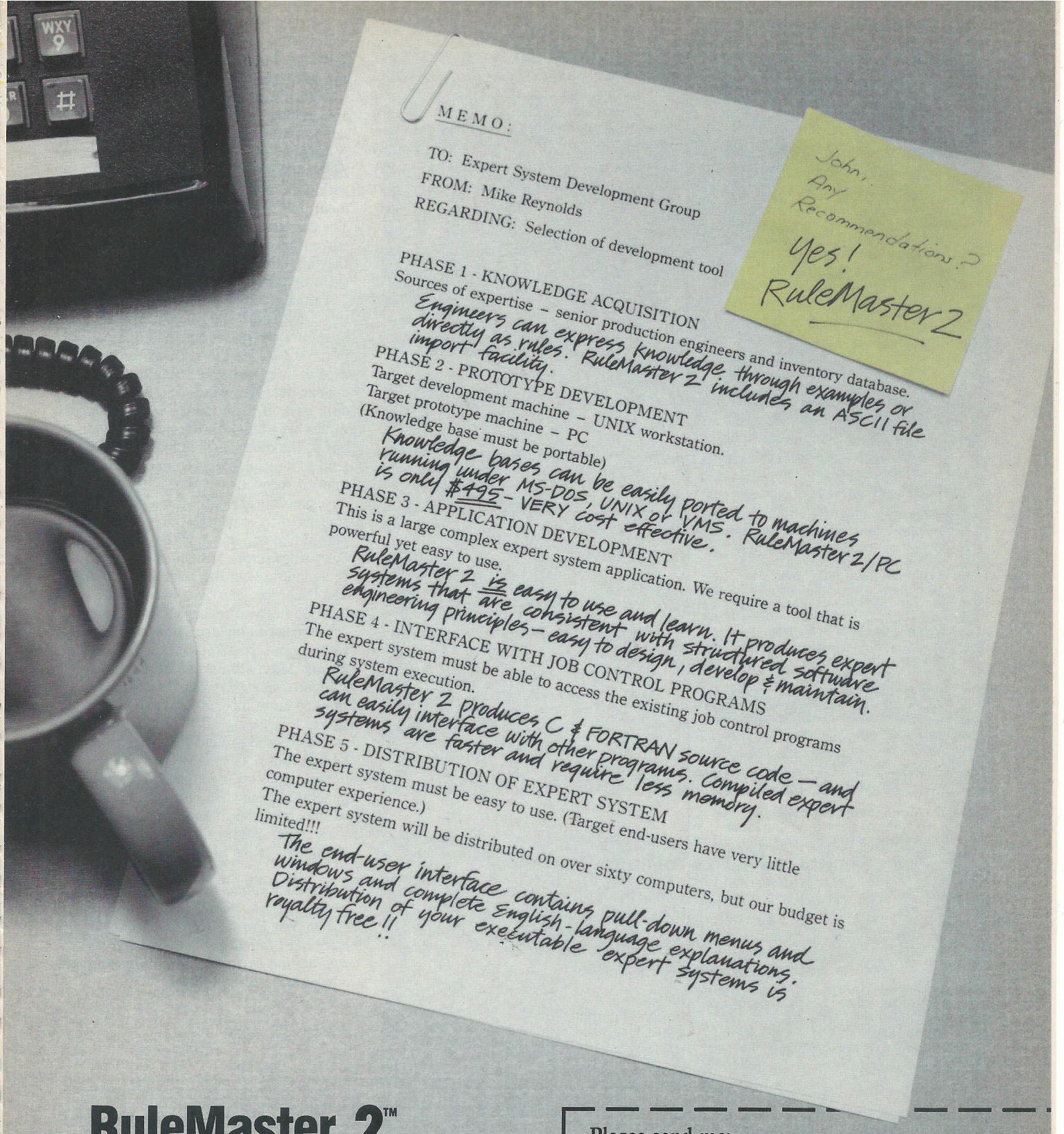
The high-level output of the screen generator includes escape's from the main entry portion of the resulting program in the form of the *ENTER.SAVE, ENTER.EXIT, ENTER.DELETE,* and other such statement groups. For example, just before a data base record is updated, statements in the *ENTER.SAVE* subsection of the program will be executed. These might extend to hundreds of lines of DataFlex procedural code in either the macro or the lower-level language (mixing the two is tricky), performing various operations on other files in the data base.

Using either level of the DataFlex language requires some programming skill, similar to d:BASE III Plus. Screens addressed by the code continue to be produced by a two-dimensional mapping process of editing a screen image and compiling it into the top of the program module most closely associated with the screen. Listing 2 is an example of the output of the DataFlex screen generation process.

While screen generators first appeared as part of microcomputer-based DBMSs, they are now associated with many third generation languages. These products allow programmers to paint a screen using a technique similar to R:base or DataFlex and turn the screen into its equivalent in third generation code (for example, COBOL, BASIC, or C). The main difference between these products and the screen generators of the DBMSs is that these products cannot automatically associate a screen window or field with a particular file or field from a table.

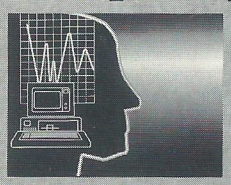
Cincom's Mantis screen generator is like a general-purpose screen generator, with a few interesting twists. Mantis is a high-level procedural and interpreted language that runs under CICS on IBM mainframes and Digital Equipment Corp.'s VMS on Vaxen.

The Mantis screen generator allows a designer to paint a screen and set up certain parameters for acceptable values in each field (these facilities are limited compared to those of R:base, Informix, or Revelation). It allows a designer to specify a number of buffers for a screen



RuleMaster 2™

the right tool for all phases of expert system development



RADIAN CORPORATION

(512)454-4797

CIRCLE 23 ON READER SERVICE CARD

A company of The Hartford Steam Boiler Inspection and Insurance Company

RuleMaster 2 is a trademark and RuleMaster is a registered trademark of Radian Corporation.
MS-DOS is a registered trademark of Microsoft Corporation, UNIX is a registered trademark of AT&T Bell Laboratories, and VMS is a registered trademark of Digital Equipment Corporation.

Please send me:

☐ additional information on RuleMaster 2
☐ a RuleMaster 2 demonstration disk ($10.00) ☐ 5¼" ☐ 3½"

Payment: check ☐ VISA ☐ Mastercard ☐

Card No. _____ Exp. Date _____

Signature _____

Name _____

Address _____

City _____ State _____ Zip _____

Phone _____

Send to: RuleMaster® / #270 / P.O. Box 9802 / Austin, TX 78766-0802

CL

```
1  OPEN CUSTOMER
2  OPEN ORDERS
3
4  TODAYS_DATE=TODAY()
5
6  ENTER CUSTOMER
7    ENTRY X.ODR_DATE      [NOENTRY]
8    ENTRY X.CUST_NBR      [CAPSLOCK,AUTOFIND,FINDREQ]
9    ENTRY X.ACTV_ODR_NBR  [REQUIRED]
10   ENTRY X.LAST          [NOENTRY]
11   ENTRY X.MI            [NOENTRY]
12   ENTRY X.FIRST         [NOENTRY]
13   ENTRY X.CRDT_LMT      [NOENTRY]
14   ENTRY X.SIX_MO_TOT    [NOENTRY]
15  ENTRY X.SIX_MO_PD      [NOENTRY]
16  ENTRY X.CUR_MO_TOT     [NOENTRY]
17  ACCEPT WINDOW.10       [CAPSLOCK,CHECK="YN",REQUIRED]
18
19 ENTER.SAVE
20   IF SENSITIVE_INFO.10 EQ "Y"
21     ORDERS.ODR_NBR=X.OPER_ODR_NBR
22     FIND ORDERS BY INDEX.1
23     [FOUND] BEGIN
24        MOVE SENSITIVE.INFO.10 TO ORDERS.OK_FLAG
25        MOVE TODAYS_DATE TO ORDERS.OK_DATE
26        SAVE ORDERS
27     END
28     [FINDERR] MOVE "CANNOT LOCATE PROPER ORDER" TO SENSITIVE_INFO.11
29     CLEAR ORDERS
30  RETURN
31
32  ENTER.DELETE
33  // commands executed upon delete request...
34  RETURN
35
36  ENTER.EDIT
37  // commands executed upon edit or delete request...
38  RETURN
39
40  ENTER.CLEAR
42  // commands executed upon a clear screen request
43  RETURN
44
45  ENTER.EXIT
46  // commands to execute upon exiting the entry screen...
47  RETURN
48  ENTEREND
49  ABORT

        *                   *                   *
```

Listing 2. *(Continued from page 30)*

Products mentioned in this column

DataFlex
Data Access Corp.
14000 S.W. 119 Ave.
Miami, Fla. 33186
(305) 238-0012

Informix
Informix Software Inc.
4100 Bohanon Drive
Menlo Park, Calif. 94025
(415) 322-4100

Mantis
Cincom Systems Inc.
2300 Montana Ave.
Cincinnati, Ohio 45211
(513) 662-2300

Paradox
Ansa Software
1301 Shore Way Rd. Ste. 221
Belmont, Calif. 94002
(415) 595-4469

R:Base System V
Microrim
3925 159th Ave. N.E.
Redmond, Wash. 98052
(206) 885-2000

Reflex
Borland International
4584 Scotts Valley Dr.
Scotts Valley, Calif. 95066
(408) 438-8400

UNIFY
Unify Corp.
3870 Rosin Ct.
Sacramento, Calif. 95834
(916) 920-9092

field where multiple records of a table can be displayed at the same time.

Mantis's generated screen is named in a procedure that references it with a screen statement, *SCREEN LOCAL.NAME(LIBRARY:LIBRARY.NAME, LIBRARY.PASSWORD)*. It is used within the procedure via a converse statement, *CONVERSE LOCAL.NAME*. The converse instantly displays all of the current values for the screen fields as they are stored in the buffers set up for them by the screen generator.

Values must be moved to and from these fields with the usual assignment operators unless the names of the fields in the screen are the same as those of the external view or views being used to exchange data between the screen and the data base. If these names are the same, values are automatically mapped from one to the other. This mapping can extend to multiple views from one screen or many screens to one view.

A procedural program may have many screens in its map set, which can be displayed like pop-down or tile-style windows since screen size (lines by columns) is configurable in the generation process and the screen can be conversed at any location in the physical screen space.

Screens can be larger than the physical screen (up to 250 lines by 250 columns). Mantis comes equipped with standard, single-keystroke scrolling options so large reports can be displayed on line in a large logical page that the operator can scroll around for query or update purposes.

Next month I'll compare some report generators, and then begin to explore the various procedural languages associated with contemporary DBMS products and the design issues they raise.

Matthew Rapaport is currently employed by the AMEX Life Assurance Co. in San Rafael, Calif., and has worked in fourth generation language application environment development for six years.

MICROSOFT LANGUAGES NEWSLETTER VOL. 2, NO. 11

Dear Reader:

When we introduced the Microsoft® Languages Newsletter in 1985, we had just one goal in mind: improving communications with the people who use Microsoft languages.

We wanted to communicate to you the technical advances underway at Microsoft and get feedback from you about how we could make additional improvements to our products.

Everything we do at Microsoft is built around a vision: to create the software that puts a computer on every desk and in every home. We know that many of you share that vision with us and are working toward it in the development of your own programs.

We know that in order to make this vision come true, we need to supply you with effective development tools—such as the Microsoft CodeView® debugger, Microsoft QuickBASIC and QuickC™ programming languages, and Microsoft C Optimizing Compiler.

But we also know that we need your help to achieve our vision of the future. That's why, back in January, we asked you to share your visions of the ideal programming environment. Your response was tremendous. In fact, we're already exploring many of your ideas, including language enhancements to support windowing environments, a general programmer's editor, and a "super" MAKE facility.

These are just a few examples of how your ideas and suggestions have helped us shape our future product plans. From the very beginning of the Languages Newsletter, your feedback has been invaluable. And that's why we want you to be part of some important changes.

As we look to the next couple of years, we recognize that developers will want to create more global solutions; solutions that will encompass languages, operating environments, and networking. With that in mind, we have decided to transfer our resources from the Languages Newsletter to the Microsoft Systems Journal*—a bimonthly publication covering the broad scope of systems-related issues.*

Because the Microsoft Systems Journal *will include much of the information you're used to getting from the Languages Newsletter, and because we value your active participation in shaping future products, we'd like to offer you a free issue of the* Microsoft Systems Journal.

The issue you'll receive contains detailed information about Microsoft Operating System/2 and the Microsoft OS/2 presentation manager. It gives you the inside story about where we're going, including an interview with Gordon Letwin, one of the chief architects of Microsoft OS/2. There's also a special section entitled "Ask Dr. Bob," with detailed Q&A's on subjects ranging from Microsoft Windows printer drivers to using Microsoft C math library functions in an assembly program.

To arrange to get your OS/2 issue and the Microsoft Systems Journal *on a regular basis, just call toll free 1-800-533-6625 (in Ohio, call 1-800-633-3157). We'll sign you up for the OS/2 issue plus six regular issues (a year's subscription) at the special introductory price of $34.95.* If after examining the OS/2 issue you decide you don't want to continue your subscription, just write "Cancel" across the invoice, return it to us, and you can keep the OS/2 issue free of charge.*

Thanks for your support of the Microsoft Languages Newsletter. We hope you'll find the Microsoft Systems Journal *even more valuable in your programming efforts.*

Sincerely,

Bill Gates

Bill Gates
Chairman of the Board
Microsoft Corporation

**This offer is valid only in the USA and expires March 31, 1988.*

Microsoft and CodeView are registered trademarks and QuickC is a trademark of Microsoft Corporation.

Pattern Matching Alternatives:

Theory vs. Practice

By Jack Purdum

Sooner or later, every programmer needs a routine that searches a large sequence of characters (perhaps an array or a file) for a match on a specific character pattern. I've needed such a routine many times over the years but never seemed to have the time to investigate the problem properly. I would just hack a brute force solution together until it worked. The old adage "If it ain't broke, don't fix it" tended to rule things.

Some time ago, I found need for such a routine again. This time, however, I thought I'd dig around for something a little more sophisticated than my earlier brute force algorithms. Although several algorithms could be used, I implemented three in C and performed some simple tests to evaluate their relative performance.

In the course of my search, I found an excellent article by Richard Wiggins and Paul Wolberg on string searching using the Boyer-Moore algorithm ("Searching for Strings with Boyer-Moore," *COMPUTER LANGUAGE,* Nov. 1986, pp. 28-42). Wolberg was kind enough to send me his C translation of code used in that article, and some of Wiggins' and Wolberg's work is presented here. However, I cannot do justice to their in-depth explanation of the Boyer-Moore algorithm. You owe it to yourself to read the article.

The remainder of this article presents an idea or two designed to wring a little more performance from the algorithms studied. The results of those efforts are also discussed.

Algorithms tested

There is a wealth of information on string searching. Many algorithms could have been selected for the comparison in this article, but I tested only three: Knuth-Morris-Pratt, brute force, and Boyer-Moore.

It became obvious at an early stage in the testing that the Knuth-Morris-Pratt algorithm could not beat Boyer-Moore in performance on any reasonable pattern. For the patterns tested, Boyer-Moore was at least twice as fast and in some cases beat Knuth-Morris-Pratt by a factor of 9. (If you are interested in knowing more about the Knuth-Morris-Pratt algorithm, a good discussion is found in

Robert Sedgewick's *Algorithms*, Addison-Wesley, 1984, pp. 241-254. Chapter 19 of this book is especially useful.)

Using brute force

Because I had always used brute force for pattern matching in the past, the brute force algorithm is presented as a basis from which the performance of the other algorithms is measured.

A brute force approach to string searching involves simply looking at each character in the string and comparing it to the first character in the pattern. If the first character matches, the next character in the string is compared against the next character in the pattern. Eventually, you either force your way through the string or find a match on the pattern.

As was pointed out in the Wiggins and Wolberg article, the brute force algorithm is a function of the number of characters in the pattern (m) and the number of characters in the string being searched (proportional to $m*n$). Textbooks often present the theoretical timing function for a brute force search algorithm as $O(m*n)$. This approach obviously suggests that the brute force algorithm degrades as m and/or n increase.

The first version of the brute force algorithm is the one used by Wiggins and Wolberg to gauge the performance of the Boyer-Moore algorithm. Their code appears in Listing 1. Most of the original comments have been stripped from the code. Even so, the code is pretty easy to understand.

The Wiggins and Wolberg brute force algorithm (called WWBF from now on) is based upon the assumption that *MAXLINE* lines of text of *MAXSTR* length are read into the character array to be searched (*Target*). Variable *NumFound* keeps track of the number of matches located in the search string.

The *posBF()* function does the real work. The brute force algorithm simply looks at a character in the target string and tests for a match on the first character in the pattern being searched. If the character in the target string matches the first character in the pattern, a second *for* loop checks for additional matches between the target and pattern strings. If we continue in the inner *for* loop *Patlen*−*1* times, we have a match, and *posBF()* returns the index into the target string.

Because I needed a memory-based pattern algorithm on data that might not contain any *newline* characters in the file, my test code differed from that used by Wiggins and Wolberg. Also, their test code is designed to count the number of matches found in a file.

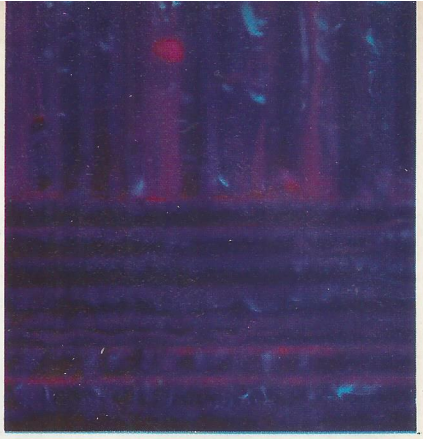
My design goal was to get a pointer to the match, if one is found. I modified Wiggins' and Wolberg's code to return pointers and then tested their brute force implementation using my test data files. In my version of the test code, a data file is opened, its size determined, and then

```
Wiggins-Wolberg brute force

/*
        TargetList -- an array of strings to search.
*         NumLines -- the number of elements in Target.
*          Pattern -- the pattern to search for.
*/
DoBruteForce(TargetList, NumLines, Pattern, NumFound)
char TargetList[MAXLINE][MAXSTR+1];
int NumLines;
char Pattern[];
int *NumFound;
{
     int  PBF;  /* Position of Pattern in Target
                   as returned by posBF */
     int  i;    /* Loop counter */
     for (i = 0; i <= NumLines; i++) {
        PBF = posBF(TargetList[i], Pattern);
        if (PBF > 0){
           (*NumFound)++;
        }
     }
}
posBF(Target, Pattern)
char Target[];   /* String to search. */
char Pattern[];  /* String to search for. */
{
     int  Stop;   /* The last position in Target
                     to look for Pattern */
     int  i;      /* The current position in Target for search */
     int  j;      /* The current position in Pattern being compared
                     to the corresponding position in Target */
     int  Patlen;
     Patlen = strlen(Pattern);
     Stop = strlen(Target) - Patlen;
     for (i = 0; i <= Stop; i++){
        for (j = 0; Pattern[j] == Target[i + j]; j++){
           if (j == Patlen-1) return(i+1);
        }                                    /* end for j */
     }                                       /* end for i */
     return(0);
}
```

Listing 1.

```
Modified WWBF algorithm

/*
*********************************************************************
*                    D O B R U T E F O R C E
      Code supplied by Paul Wolberg. Modified by JJP to return
      a pointer to match or NULL if no match.
*********************************************************************/
char  *DoBruteForce(char Target[], char Pattern[], int n)
{
   int Patlen,     /* The length of Pattern */
       i,          /* The current position in Target for search */
       j;          /* The current position in Pattern compared
                      to the corresponding position in Target */
   Patlen = strlen(Pattern);
   i = 0;
   for (;;){
       if (Pattern[0] == Target[i]) {
          if (strncmp(Pattern, &Target[i], Patlen) == 0)
             break;
       }
       i++;
   }
   if (i > n - Patlen) {
      return NULL;              /* Pattern not found in Target. */
   } else {
      return &Target[i];
   }
}
```

Listing 2.



enough memory dynamically allocated to read the file into memory. As explained in the next section, I also modified the algorithm to use a sentinel value in the search process. (More on sentinels in a moment.)

The test environment was an AT-type machine running at 7.2 MHz with no wait states. All code was compiled with the Ecosoft Eco-C88 C compiler, although any C compiler should handle the code with little or no modification.

Due to the resolution of the system clock, each search is repeated 100 times. That is, we search for a pattern in a file 100 times to get the times high enough so we can use the MS-DOS system clock for timings. All of the test files are approximately 27.3K in length, and the search patterns are the last item in the file. That is, if we are going to search for the name Mary, we will look through approximately 27.3K of text before finding a match. Therefore, the timings are based upon searching about 2.73MB of text with each of the algorithms tested.

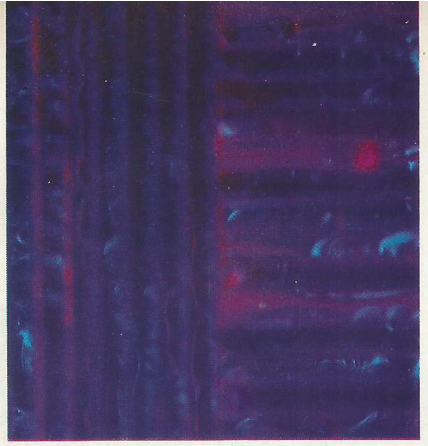
Sentinels

After modifying the WWBF algorithm for use with my testing scheme, the test time on one of the data files was about 41 seconds. I then changed Wiggins' and Wolberg's code slightly, as shown in Listing 2.

The careful reader might look at Listing 2 and say it won't always work properly because the function does not check to see if there are any more characters to search in the *Target[]* character array. Well, you're sorta right.

What is not obvious is the use of what is called a sentinel. The program devised to test the algorithms (Listing 3) first asks for the pattern to be searched for or *Pattern[]* in the argument list for the brute force search.

Next, the program finds out how large the test file is (three files were used for testing each algorithm). The program then requests sufficient storage to hold the entire file in memory *plus* a copy of the pattern string being searched. The program then reads the file into memory and adds the pattern string to the end of the file in memory. This copy of the



search pattern at the end of the file is called the sentinel.

Think about what the sentinel does for us. We now know with certainty that any search will at least find a match on the sentinel that's been added at the end of the text file. In most algorithms, this means we can eliminate the test to see if we have searched the entire text string. In our test procedure, using the sentinel saves about 2.73 million comparisons. A million here, a million there...they all add up.

Only one problem remains: we must be able to distinguish between a match on the sentinel and a "real" match in the text being searched. The problem is easily solved: if the match occurs after looking at more than *strlen(Target)* characters, we know the match was on the sentinel. If you look at how the return values are determined in the function, you will see how we detect whether we match on the sentinel or on a real match:

```
if (i > n - Patlen) {
   return NULL;   /*Pattern not in Target*/
} else {
   return &Target[i];
}
```

If *i* is greater than the length of the search string minus the length of the pattern, we matched on the sentinel value rather than a pattern in the target string. Therefore, we return a *NULL* pointer to indicate no match. Otherwise, we found a real match in the string and return a pointer that points to the match in the target string.

Two other changes appear in Listing 2. First, we removed the overhead associated with a function call by moving the WWBF *posBF()* function call into *DoBruteForce()*. Second, we used the standard library function *strncmp()* to perform the check for a pattern match. This allows us to remove one *for* loop and let *strncmp()* do most of the work. As a result of the changes reflected in Listing 2, one of the search times decreased from 41 seconds to less than 28 seconds.

The row labeled "Modified WWBF"

C source code for test program

```
#include <stdio.h
#include <string.h>
#include <malloc.h>

#define MAXCHAR 129

#ifndef TRUE
   #define TRUE    1
#endif

char *getmemory();
long clock(), dosize();
void copyfile(), dofile(), dotest();

FILE *fp;

int main(int argc, char **argv)
{
   char pat[MAXCHAR], *ptr, *start;
   int big, i, m, n, next, which;
   unsigned int total;
   long b;

   total = getpat(pat);                    /* Get pattern and length     */
   dofile(argv);                           /* Open the file--set fp      */
   b = dosize();                           /* How big is the file        */
   total += (unsigned) b + 1;              /* Enough for file + pattern  */
   start = ptr = getmemory(total);         /* Get the memory             */
   copyfile(ptr, pat);                     /* Copy file to memory        */
   ptr = start;
   which = select();                       /* Which test                 */
   dotest(start, pat, which, (int) b); /* Run a test        */
}
/*************************************************************
                        dotest()
      Find out which test to run and run it.
Args:
      char *start = start of characters to be searched.
            *pat = pattern to look for
        int which = which test to run.
              big = size of the file.
Return value:
      none.
*************************************************************/
void dotest(char *start, char *pat, int which, int big)
{
   char *brut(), *brut2(), *boy_m(), *kmp(), *DoDelta2(), *DoDelta1();
   char *ptr, *DoBruteForce();
   int i;
   long t;
   double time;
   switch(which) {
      case 1:                          /* Sedgewick's Boyer-Moore */
         clock();
         for (i = 0; i < 100; i++)
            ptr = boy_m(start, pat, big);
         t = clock();
         break;
      case 2:                          /* Fast brute force */
         clock();
         for (i = 0; i < 100; i++)
            ptr = brut(start,  pat, big);
         t = clock();
         break;
      case 3:                 /* Wiggins and Wolberg brute force */
         clock();
         for (i = 0; i < 100; i++)
            ptr = DoBruteForce(start, pat, big);
         t = clock();
         break;
      case 4:                          /* WWBM with Delta2 */
         clock();
         for (i = 0; i < 100; i++)
            ptr = DoDelta2(start, pat, big);
         t = clock();
         break;
      default:
         t = 0L;
         break;
```

Listing 3. *(Continued on page 38.)*

```
    }
    time = (double) t / 1000000.0;      /* Get down to seconds */
    if (ptr == NULL)
       printf("No match.");
    else
       printf("\n\nptr=%s", ptr);
    cursor(22, 10);
    printf("Search Complete. Time = %g sec (%ld clock ticks)", time, t);
}
/*************************************************************
                          select()
      Which test will be run.
Args:          void
Return value:  int  selection value.
*************************************************************/
int select(void)
{
    char buff[10];
    int i;

    clrscr();
    printf("                    Which test:\n\n");
    printf("1. Sedgewick's B-M   2. Brute Force, V.2      3.W-WBF     ");
    printf("4. W-W w/Delta2\n");
    for (;;) {
        printf("\nEnter choice: ");
        i = atoi(gets(buff));
        if (i > 0 && i < 5)
           break;
        else
           printf("\a");
    }
    return i;
}
/*************************************************************
                              copyfile()
      Copy test file into memory. Note a sentinel copy is also
    added.
Args:          char *ptr = where the file goes in memory.
               *pat = the pattern to locate.
Return value:  none.
*************************************************************/
void copyfile(char *ptr, char *pat)
{
    int i;

    while ((i = fgetc(fp)) != EOF) {    /* Copy file to memory  */
       *ptr++ = i;
    }
    for (i = 0; pat[i]; i++)            /* Add sentinel at end  */
       *(ptr + i) = pat[i];             /* See text.            */
    fclose(fp);
}
/*************************************************************
                            getmemory()
      Get some memory from the op system.
Args:          int t = number of bytes needed.

Return value:  char * -- a pointer to t bytes of memory.
*************************************************************/
char *getmemory(int t)
{
    char *ptr;
    ptr = calloc(t, sizeof(char));
    if (ptr == NULL) {
       printf("Not enough memory");
       exit(1);
    }
    return (ptr);
}
```

Listing 3. *(Continued on page 39.)*

in Table 1 uses the code from Listing 2 as the basis for testing.

All of the algorithms implement a sentinel. I did this because the sentinel didn't affect my design goals, was easy to code, added very little overhead, and reduced the search time by about 20% in most cases. Also, it is not difficult to remove the sentinel if desired.

Brute force, version 2

Flushed with the success of the modified WWBF algorithm, I tried to think of more efficient ways to implement the brute force algorithm. After some trial-and-error fiddling, the code in Listing 4 emerged.

The primary difference between WWBF and version 2 is the use of the *memchr()* and *memcmp()* functions. Many C compilers have these functions (the functions are part of both UNIX System V and the pending ANSI X3J11 definition for the standard C library). The advantage of these functions is that they are faster than their *str* counterparts *strchr()* and *strncmp()* because they are simple and do less work (they don't know about nulls or anything else).

The *memchr()* function is designed to examine up to *n* characters for a match between the character pointed to by *t* and *p*. If a match is found, *memcmp()* is called using the pointer returned by *memchr()* as the index into the target string and comparing it with the pattern string pointed to by *p*. The *memcmp()* function then examines *n* (the pattern length) characters to see if a match is found. If a match is found, we break out of the infinite *for* loop, check for a real match vs. the sentinel, and return the appropriate pointer. If no match is found by *memcmp()*, we up the pointer in the target string and continue looking. Ultimately, we must either find a match or the sentinel and appropriate pointer is returned.

Version 2 of the brute force algorithm reduced the test time to just under 3.5 seconds. We now have an implementation of the brute force algorithm that is simple to understand, uses standard C library routines (so it should be fairly portable), has pretty small code size, and is

over 10 times faster than our original version. The brute force presented in Listing 4 appears as the last row in Table 1.

Boyer-Moore, version 1

There is little room to improve on the discussion of the Boyer-Moore algorithm as presented by Wiggins and Wolberg. I encourage the reader to study their examples.

One important distinction pointed out in the Wiggins and Wolberg article is the linearity of Boyer-Moore. That is, its performance is a function of $m+n$ rather than $m*n$, as was the case for the brute force algorithm (a theoretical timing function of $O(m+n)$). What follows in this section is just enough information to give you some idea of how the algorithm works.

The Boyer-Moore algorithm decides where to continue searching based on any mismatch on the last character of the pattern and the string being searched. For example, if we are looking for the pattern hooli in the sentence:

```
hoola hoola girls like hoopla and hooligans.
(1) hooli
(2)    hooli
(3)     hooli
```

the mismatch is on the a in hoola and the i in hooli. Upon finding a mismatch, the

```
/*****************************************************************
                            dosize()
      How large is disk file? Need for dynamic memory allocation.
Args:          none.
Return value:  long -- size of data file to be read into memory.
*****************************************************************/
long dosize(void)
{
   long b;
   fseek(fp, 0L, 2);                /* Go to end of file    */
   b = ftell(fp);                   /* See how big it is    */
   fseek(fp, 0L, 0);                /* Rewind to the start  */
   return (b);
}
/*****************************************************************
                            dofile()
      Open a disk file. Note use of binary flag to get around
      carriage-return line-feed adjustments.
Args:          int argv = name of file from the command line.
Return value:  none.
*****************************************************************/
void dofile(char **argv)
{
   if ((fp = fopen(argv[1], "rb")) == NULL) {
      printf("Can't open it.\n");
      exit(1);
   }
}
/*****************************************************************
                            getpat()
      What is the search pattern?
Args:          char *pat = character array for search pattern.
Return value:  int -- length of the search pattern.
*****************************************************************/
int getpat(char *pat)
{
   clrscr();
   printf("\n\nEnter search string: ");
   gets(pat);
   return ( strlen(pat) );
}
```

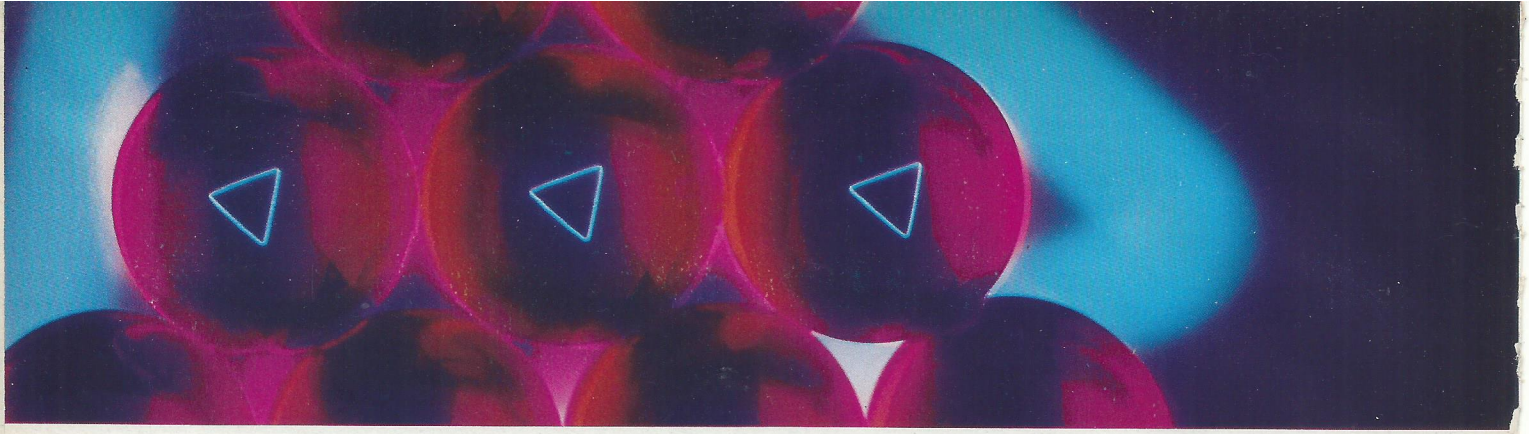
Listing 3. *(Continued from page 38.)*

Indexed test results (sec)

	Test										
Algorithm	Big Mary	Little Mary	abc	pabc	hooli	string	computer	table	zeeee	eeeeeeez	Average
Modified WWBF	39.9	39.7	80.9	41.3	53.6	41.0	41.2	41.4	39.9	30.4	44.9
Boyer-Moore, version 1	6.1	28.9	8.8	72.0	27.5	21.1	16.7	24.8	23.9	15.0	24.5
WWBM with Delta2	2.1[1]	7.3	2.7[1]	23.8	7.4[1]	5.5[1]	4.6[1]	6.9[1]	6.4	3.8[1]	7.1
Brute force, version 2	3.2	3.3[1]	14.8	3.2[1]	13.8	6.2	6.1	7.0	3.2[1]	7.8	6.9

1. Fastest time for the test.

Table 1.



algorithm reads an array (configured prior to the text search) to determine how many characters can be skipped in the string. Because the a is not duplicated elsewhere in the pattern string, we can skip five characters before trying another match. Therefore, the second text compares the l in the string with the i in the pattern. This time, however, we can only advance one position because of the l in the pattern.

In case 3, we again fail on the a in the string. Because the a does not appear in the pattern, we can advance *strlen(patern)* characters for the next test. Clearly, the Boyer-Moore algorithm is going to allow us to skip along quite fast through the text, especially if the last character (for example, a hyphen) in the pattern is not used very often in the string being searched or the last character of the pattern is not repeated in the pattern.

My first attempt at coding the Boyer-Moore algorithm (Listing 5) was taken from the algorithm presented in Sedgewick's text. Note how the *memset()* call is used to set all elements of the *delta1[]* array to the length of the pattern string. The *for* loop is then used to change only those elements that appear in the search string. The values in the *delta1[]* array determine how far we can safely skip in the target string if a match fails. Also note that we have added a sentinel, and the final *if* statement tells us if we have a real match or if we found the sentinel.

Note that the Boyer-Moore performance is significantly influenced by the pattern being searched. For example, the Big Mary test in Table 1 had the sentence "Mary had a little lamb whose fleece was white as snow." as the last item in the file. A search on just the word "Mary" took about 28.9 seconds, while a search on the entire sentence took only 6.2 seconds. Part of the improvement is because a search on the entire sentence is based on a period as the last character in the pattern, a character not replicated in the pattern string. This, plus the fact that the pattern is quite long, allows us to skip through the text string very rapidly. This is where Boyer-Moore shines: relatively large values for *m*.

I shall have more to say about pattern sensitivity in the section that discusses the test results. The code presented in Listing 5 is the Boyer-Moore version 1 that appears in Table 1.

```
Brute force, version 2

/*****************************************************************
                              brut()
     About as fast as we can do a dumb brute force search in C.

Args:              char a[] = string to search
                        p[] = pattern to search for in a[]
                     int n = length of a[].

Return value:      char *    a pointer to the match if found,
                             NULL if no match.
*****************************************************************/
char *brut(char a[], char p[], int n)
{
   char *ptr, *s;
   int m;
   s = a;
   m = strlen(p);
   for (;;) {
      if ((ptr = memchr(a, p[0], n)) == NULL)
         break;
      if (memcmp(ptr, p, m) == 0)
         break;
      a = ptr + 1;
   }
   return ((n - (ptr - s) <= 0) ? NULL : ptr);
}
```

Listing 4.

Boyer-Moore, version 2

The Wiggins and Wolberg article published a second version of the Boyer-Moore algorithm (Listing 6) that uses a second Delta table. While that article explains this refinement in detail, the purpose of the *delta2[]* array is to allow the largest skip value possible on a false match while ensuring that a match is not missed. Because the *delta2[]* array uses substrings to determine maximum skip values, this refinement performs best when the pattern string is repetitive.

Although I made some modifications to the code to conform to my testing procedure, the code shown in Listing 6 is essentially the code provided by Wolberg. While the authors were apologetic about the *GOTO*'s in their article, perhaps they shouldn't have been. I rewrote the code using infinite *for* loops and a sentinel and it actually ran slightly slower than their version. (Nobody said *GOTO*'s were inefficient.) Therefore, the code in Listing 5 is very true to the Wiggins and Wolberg implementation of Boyer-Moore, with the Delta2 refinement.

Test data

As already mentioned, the source code for the test program appears in Listing 3. The program allows the test file to be specified as part of the command line.

Three test files were built, each about 27,300 bytes long.

The first test file was several C source code programs added end-to-end with one line added as the last line in the file: "Mary had a little lamb whose fleece was white as snow." This sentence was the search pattern for the Big Mary test. A second test simply looked for the single word "Mary" and is called the Little Mary test.

The Mary test file also contains several common words added at the end of the file. These words were also part of the test results presented in the Wiggins and Wolberg article. Table 1 shows the words used.

The second file, the hooli test, contained the following sentence:

```
hoola hoola girls like hoopla and hoolagans.
```

repeated enough times to build a file of approximately 27,300 bytes in length. The last word in the file was then changed to hooligans (note the exchange of a for i). The search pattern for this test was hooli. This pattern was selected because it can have a fairly high number of false matches in the file. This file should favor the Boyer-Moore algorithm because it is a nonrepeating pattern within the pattern string and the text is highly repetitive.

The final test file, called the abc test, consisted of the pattern abcdefghijklm repeated enough times to build a 27,300-byte file. No *newline* characters appeared in the file. The last entry in the file had an n added to it. The search pattern was abcdefghijklmn. This file also has a high repeat factor and should favor the Boyer-Moore algorithm.

A spin-off of the abc test was to search for the pattern pabcdefghijklm. Because many matches would falsely start with the end of the pattern string before sensing the p, Boyer-Moore would be placed at a severe disadvantage. On the other hand, brute force version 2 should do very well because the only p in the file is the pattern itself. This test is called pabc. These two tests present a best-case, worst-case test of Boyer-Moore.

```
Boyer-Moore algorithm, version 1

/*****************************************************************
                                        boy_m()
    Boyer-Moore search based upon Sedgewick's algorithm.
Args:
   char a[] = string to search
      pat[] = pattern to search for in a[]
      int n = length of a[].

Return value:
   char * -- pointer to the match if found, NULL if no match.
*****************************************************************/
char *boy_m(char a[], char pat[], int n)
{
   unsigned char deltal[127];
   int j, offset;
   int c, i, m, term;
   m = strlen(pat);
   memset(deltal, (unsigned char) m, 127);
   for (j = 0, i = m - 1; i >= 0; i--, j++) {
      c = pat[i];
      if (deltal[c] == m)
         deltal[c] = j;
   }
   i = m - 1;
   j = m;
   while (TRUE) {
      if (a[i] == pat[j - 1]) {     /* If we have a match...   */
         i--;
         j--;
      } else {                      /* No match yet. March on  */
         i += m - j + 1;
         j = m;
         term = m - j + 1;
         offset = deltal[a[i]];
         if (offset > term) {
            i += offset - term;
         }
      }
      if (j < 1) {                  /* If j == 0; a match      */
         break;                     /* If i > n, no more string */
      }
  }
  if (i > n - m)                    /* Found sentinel   */
     return NULL;
  else
     return &a[i + 1];
}
```

Listing 5.

```
WWBM with Delta2

/*
*********************************************************************
*                          D O D E L T A 2

      Code supplied by Paul Wolberg. Modified by JJP to return
      a pointer to match or NULL if no match.
*********************************************************************/

char *DoDelta2(char Target[], char Pattern[], int n)
{
   char *posDel2();
   int Deltal[ASCIILEN+1],        /* The deltal table. */
       Delta2[MAXSTR+1],          /* The delta2 table. */
       patlen;                    /* Length of Pattern. */
   MakeDell(Pattern, Deltal);
   patlen = strlen(Pattern);
   Deltal[Pattern[patlen - 1]] = LARGE;
   MakeDel2(Pattern, Delta2);     /* Create the delta2 table. */
                                  /* ShowDel2(Pattern, Delta2); */
   return posDel2(Target, Pattern, Deltal, Delta2, n);
}
/*
*********************************************************************
*                          P O S D E L T A 2
*********************************************************************/
char *posDel2(char Target[], char Pattern[], int Deltal[],
                                             int Delta2[], int n)
{
   int Trglen,     /* The length of Target     */
       Patlen,     /* The length of Pattern    */
       j,          /* Loop counters            */
       hop;        /* Number of characters to move along
                      Target before next comparison.        */
   int i;
   Trglen = n;
   Patlen = strlen(Pattern);
   if (Patlen > Trglen)
      return (NULL);
   i = Patlen - 1;

FAST:
   i = i + Deltal[Target[i]];
   if (i > 0)
   if (i < LARGE)
      return (NULL);
   i = (i - LARGE) - 1;
   j = (Patlen - 1) - 1;

SLOW:
   if (j < 0 && i < n - Patlen)     /* Modified for sentinel   */
      return (&Target[i + 1]);
   if (Target[i] == Pattern[j]) {   /* Found match on character */
      j--;                          /* Get ready for next one */
      i--;
      goto SLOW;                    /* See if we've looked at all */
   }
   hop = Deltal[Target[i]];         /* No match. Set up skip value */
   if (hop == LARGE)
      hop = 0;
   if (hop > Delta2[j]) {           /* Which skip can we use? */
      i = i + hop;
   } else {
      i = i + Delta2[j];
   }
   if (i < Trglen){                 /* Have we looked at all?  */
      goto FAST;
   } else {
      return (NULL);
   }
}
```

Listing 6. *(Continued on page 43.)*

As you can see from the code in Listing 6, each test time is based on 100 complete searches of the file in memory for the pattern. Test times are an average of several runs using a *clock()* function for timing.

Fairness in testing

It can be argued that it is unfair to compare a brute force implementation using the *memchr()* and *memcmp()* functions because they are coded for speed (often in assembler) and are not part of the Boyer-Moore algorithm. One might further argue that the only fair comparison would be a brute force written in pure C.

Well, the tests aren't fair. My goal was to write a function capable of searching a string as quickly as possible given the resources at my disposal. In my mind, using efficient functions to accomplish a task makes sense. If this is unfair, so be it.

The results of the tests are shown in Table 1. The entries in the table are the number of seconds required to run 100 iterations of the search for each of the algorithms tested. The times should be viewed as good approximations since they are averages of several runs and are limited to the resolution of my system clock. However, the times do give an indication of the relative performance of the various algorithms tested.

The Big Mary test is on the string "Mary had a little lamb whose fleece was white as snow." while the Little Mary test is on the single word "Mary." All of the other tests are on the strings shown in Table 1.

The first thing you might notice is that the refinements to the first version of the brute force algorithm paid off. On average, the modified brute force is over seven times faster than the original version. Implementing a sentinel in the algorithm accounts for between 10% and 15% of the improvement.

In the Wiggins and Wolberg article, the largest difference between the brute force and Boyer-Moore algorithms was a search pattern of eeeeeeez, with Boyer-Moore winning by a factor of 10. With the new implementation, the difference is a factor of about 2. (Again, the results

are biased because of the sentinel used to improve performance.)

Next, notice that the Boyer-Moore algorithm outdistances the pack on the abc and hooli tests, both of which contain files with highly repetitive text sequences using nonrepetitive search patterns. Conversely, the algorithm does much worse on the pabc test using the same text file but a highly false search pattern. (A highly false search pattern would have matches on all characters in the pattern except the first.)

Also notice that Boyer-Moore does not do well on the zeeee test. This is due to the repetitive nature of the pattern and the high frequency of the letter e in text files.

The brute force version 2 performance is almost opposite that of Boyer-Moore on the unusual files. In particular, notice the performance difference on the pabc test. Brute force version 2 does well because there is only one p in the file so *memchr()* spins through the file very quickly. Boyer-Moore, on the other hand, has to execute a lot of false test code (because of matches on abcdefghijklm) before finding out it's not a match on the letter p.

It is important to notice that Boyer-Moore and brute force version 2 have very similar times on real words. When you consider 100 passes through a 27K text file, the time difference would go unnoticed on a single pass through a file.

And the winner is...

This exercise suggests several conclusions. First, I agree with Wiggins and Wolberg that the Boyer-Moore algorithm typically outperforms a brute force algorithm. You can, however, devise pernicious data sets to show either algorithm the winner.

Second, the time difference between the Boyer-Moore and brute force algorithms is very small in many cases. (This is especially true for my problem of finding one or two standard English words in a relatively small text file.)

Third, there is a big difference between the statement of an algorithm in pseudocode and its final implementation in a working language. Indeed, in several

```
/*
*********************************************************************
*                          M A K E D E L 1

      Code supplied by Paul Wolberg.
*********************************************************************/
void MakeDel1(char Pattern[], int Delta1[])
{
   int  Patlen,              /* The pattern length */
        i;                   /* a loop counter */
   Patlen = strlen(Pattern);
   for (i = 0; i <= ASCIILEN; i++) {
      Delta1[i] = Patlen;
   }
   for (i = 0; i < Patlen; i++)
      Delta1[Pattern[i]] = Patlen - (i + 1);
}
/*
*********************************************************************
*                          M A K E D E L 2
      Code supplied by Paul Wolberg.
*********************************************************************/
void MakeDel2(char Pattern[], int Delta2[])
{
   int k,
       patlen,       /* Length of Pattern. */
       rightend,     /* The array index pointing to last
                        character in Pattern. */
       patlen2,      /* Twice the length of Pattern. */
       j, j1, f[MAXSTR], f1[MAXSTR],
       q, q1, t, t1;
   patlen = strlen(Pattern);
   rightend = patlen - 1;
   patlen2 = 2 * patlen;
   for (k = 0; k <= rightend; k++) {
      Delta2[k] = patlen2 - (k+1);
   }
   j = rightend;
   t = rightend + 1;
   while (j >= 0) {
      f[j] = t;
      while ((t <= rightend) && (Pattern[j] != Pattern[t])) {
         if (rightend - j < Delta2[t])
            Delta2[t] = rightend - j;
         t = f[t];
      }                          /* end while ((t <= */
      t--;
      j--;
   }                                        /* while (j >= 0) */
   q = t;
   t = rightend - q;
   q1 = 0;
                     /* Section labeled B1 in Rytter's paper. */
   j1 = 0;
   t1 = -1;
   while (j1 <= t) {
     f1[j1] = t1;
     while ((t1 >= 0) && (Pattern[j1] != Pattern[t1])) {
        t1 = f1[t1];
     }
     t1++;
     j1++;
   }
   while (q < rightend) {
      for (k = q1; k <= q; k++) {
         if (rightend + q + 1 - k < Delta2[k])
            Delta2[k] = rightend + q + 1 - k;
      }
      q1 = q + 1;
      q = q + t - f1[t];
      t = f1[t];
   }
}
```

Listing 6. *(Continued from page 42.)*

cases in Table 1 the language is the same and the fundamental algorithm is similar, yet the two implementations differ in performance by a factor of 10 or more. Given that a good part of the performance difference stems from using different library functions, a related conclusion is that it pays to know what your standard C library has to offer in the way of alternatives.

At the beginning of this article, I reiterated the theoretical specifications (linear vs. quadratic) given in the Wiggins and Wolberg article for the algorithms tested. The terms linear and quadratic alone suggest that the times should be substantially different. Empirically, however, they are not. The reasons for this disparity are many, such as choice of language, resources at hand, and implementation of the algorithm.

There is another factor, however: "Can I as a programmer, given my skills and tools, implement a given algorithm in a way that meets its theoretical timings?" In theory I should be able to, but in fact I doubt that I could.

Programmers may overlook the fact that theoretical timings are functions for a given set of parameters. In most cases, the theoretical function is stated as $O(n+m)$ or $O(n*m)$. Because the theoretical times are stated in functional form, we tend to immediately assume that the latter is slower than the former. Not necessarily so.

We know, for example, that Boyer-Moore requires considerable set up time (for example, the Delta tables) before it can begin processing the string. The set up time for the brute force algorithm is virtually nil.

To illustrate what this implies, let's make the (simplifying) assumption that both functions are linear in form. Let's further assume that our (hypothetical) function for Boyer-Moore has the form:

```
time=500,000+.1(n+m)
```

The equation suggests that 500,000 units of time are spent setting things up for Boyer-Moore.

On the other hand, a (hypothetical) brute force function might look like:

```
time=300+.8(n*m)
```

While Boyer-Moore has a short process time (the .1 coefficient multiplied by the linear behavior for n and m), the relatively high constant term (500,000) due to the high set up time suggests the algorithm starts slowly and gets better as n and m increase.

On the other hand, brute force has a low set up time but loses performance proportionately as n and m increase. This means brute force starts fast (good for short patterns) and fades in the stretch (not very good for long strings). If you try a value of 1 for m and 100 for n in the equations just discussed, you will see why set up time it important.

For the previous example, it should be clear that a crossover point exists where the performance of one algorithm becomes better than the other. This crossover is shown in Figure 1.

If you assume that n is constant, you can concentrate on the length of the pattern (m) being searched. If the pattern has $m1$ or fewer characters, brute force outperforms Boyer-Moore. For patterns longer than $m1$ characters, Boyer-Moore will win.

In reality, the actual functions probably don't look anything like those assumed to construct Figure 1; exact specifications of a timing function for an algorithm are rarely given. This lack of specificity often means some form of algorithm testing is necessary to see which alternative best suits your given need. (I'm assuming the function will be executed often enough to make testing worthwhile.)

The point should be clear: assuming that one algorithm will outperform another by casual inspection of their theoretical timing functions may not always be correct. Knowledge of the specifics of the function are very important to the actual behavior experienced in the trenches. You may need to test the algorithm to get some feel for its performance in your application. What looks bad in theory may be exactly what you need in practice.

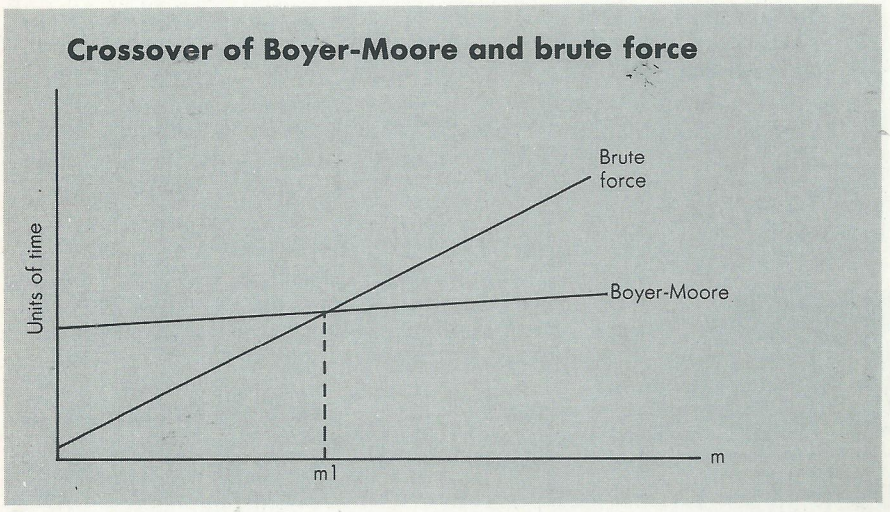
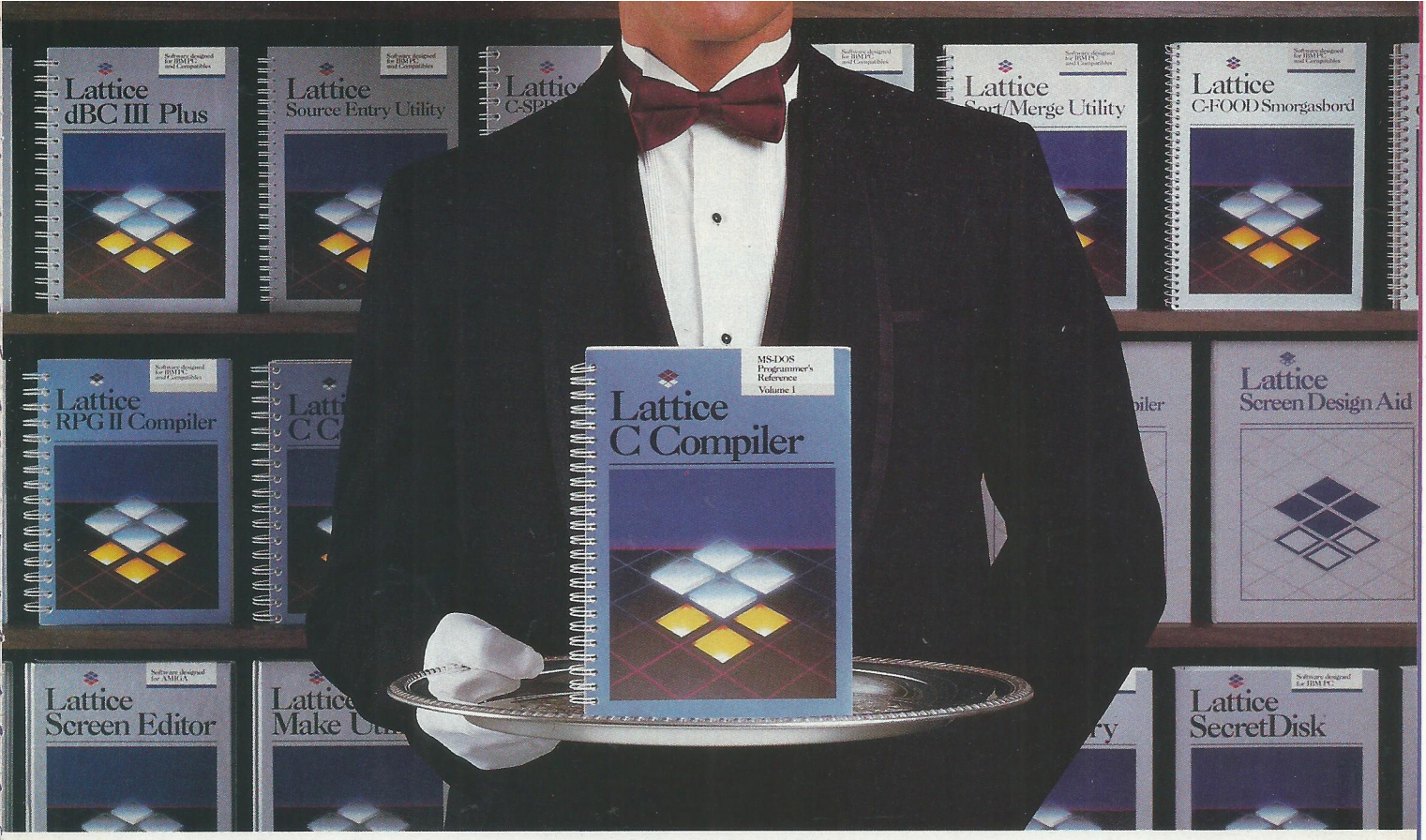


Figure 1.

Jack Purdum is a noted author of numerous C books, including C Programming Guide *and* Guide to the C Standard Library. *He is also the president of Ecosoft.*

Artwork: Anabella Gonzalez



Our software comes with something no one else can offer.

Lattice Service.

When you join the Lattice family of customers, you'll discover that your software purchase is backed by more than just an excellent warranty. It's backed by unparalleled technical support. By a total commitment to your success and satisfaction. And by Lattice's dedication to excellence in products and services.

Unlike other software manufacturers who charge you for services after you've purchased their product, Lattice offers a unique package of support programs at a price we can all live with—FREE.

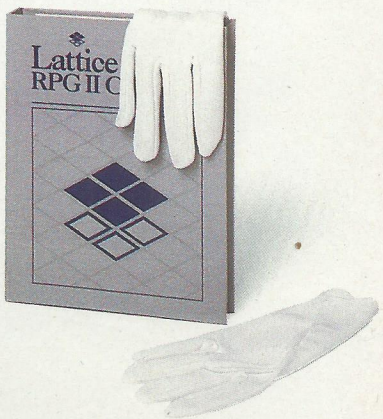
Lattice Bulletin Board Service
LBBS is our 24-hour a day bulletin board system that allows you to obtain notification of new releases, general information on Lattice products, and programs for the serious user. And if you've ever experienced the frustration of having to wait a year or more for a new release (that has corrected a bug), you'll really appreciate LBBS. Because with this service, you can actually download the latest program fixes to instantly eliminate any bugs discovered after release.

Technical Support Hotline
Responsible, dependable and capable Support Representatives are only a phone call away. You will talk to a highly skilled expert who is trained to answer any questions you have relating to specific Lattice products. Remember, your complete satisfaction is our goal.

McGraw-Hill BIX™ Network
The Byte Information Exchange (BIX) Network is a dial-in conference system that connects you with a Special Interest Group of Lattice users. The nominal one-time registration fee allows you to BIX-mail your questions—via your modem—directly to Lattice. Or you can post your questions in the conference mode for Lattice or other users to answer. Once again, you have 24-hour access.

You Also Receive:
- Timely updates and exciting enhancements
- 30-day, money-back guarantee
- Lattice Works Newsletter
- Technical Bulletins
- Access to Lattice User Groups

Lattice has developed more than 50 different Microcomputer software tools that are used by programmers worldwide. We were there for every MS-DOS release. We're there now for OS/2. And we'll be there for the next generation of technical changes. But most of all, Lattice is there for you.



Available through dealers and distributors worldwide.

Lattice

Lattice, Incorporated
2500 S. Highland Avenue
Lombard, IL 60148
Phone: 800/533-3577
In Illinois: 312/916-1600

Subsidiary of SAS Institute Inc.

CIRCLE 24 ON READER SERVICE CARD

All the Tools You Need For Motorola 680X0 From Whitesmiths

Whitesmiths, Ltd. now offers a complete set of 68K Cross Development Tools — specifically designed to work together — for the Motorola 68000 family of microprocessors. You get:

A C CROSS COMPILER
Whitesmiths' C Compilers offer the closest conformance currently available to the draft ANSI C Standard. We've added 68020 and 68881 support, and dramatically optimized code generation, so you can get the code quality you need today with the language you'll need tomorrow.

SUPPORT TOOLS
We have all the extras you need to develop embedded programs. Our powerful object utilities help you link multi-segment programs, build direct and sequential libraries, create load maps and interspersed listings, and talk to dozens of downloaders, emulators, and PROM programmers.

C SOURCE LEVEL DEBUGGING
We have the support you need to debug in terms of C functions, data types, and source lines. You debug what you write, not a lower level language.

A MICSIM SIMULATOR
You can debug your embedded programs right on your development host — our MICSIM Simulator needs no extra hardware. It's like debugging on your favorite emulator, but with no contention for dedicated resources, no download time, and with the symbolic breakpoint and trace control you've always dreamed of having.

AN XA8 CROSS ASSEMBLER
Our macro assembler is both fast and powerful, with support for 68020, 68881, and 68551.

A PASCAL COMPILER
You can program as much as you want in ISO Standard Pascal, or use the powerful extensions we've added to this production quality compiler. And you get complete integration with C and assembly language as well.

Working together, the 68K Cross Development Tools deliver both optimized performance and improved programmer productivity. Best of all, Whitesmiths offers everything you need at a very competitive price. We've been delivering and supporting high quality software development tools since 1978, and we're committed to continually enhancing our product line.

If you develop 68000 programs on a DEC VAX, an IBM PC, or a UNIX workstation, chances are we can save you time and money. For more technical details, call our toll-free number today. We also offer attractive packages for OEMs.

Whitesmiths, Ltd.
59 Power Road
Westford, MA 01886
617/692-7800

CIRCLE 25 ON READER SERVICE CARD

TOLL-FREE NUMBER 1-800-225-1030

Multiple Word Searches with LFIND

By E.J. Smith

The DOS *FIND* command can be used to search a text file for a word. Although the *FIND* command is fast, it has a few disadvantages: it is case sensitive (for example, *WRITELN* is not the same as *writeln*), and it can only search for a single word during its scan through the text file. When *FIND* locates the word, it does not indicate the line number in the file where the word was found.

The LFIND (List-find) program (Listing 1) described here accepts a list of words and searches for all of them in a single pass.

LFIND:

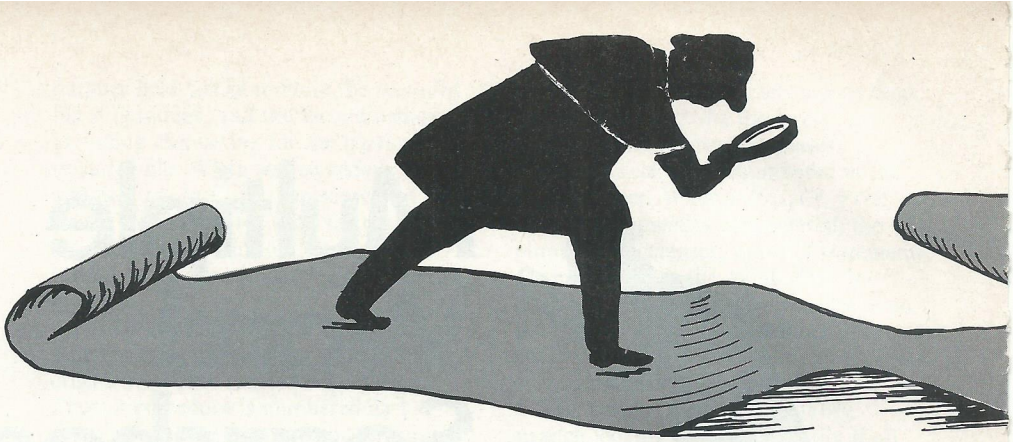
- Accepts multiple search words from a file
- Is not case sensitive
- Displays each line where a word is found
- Shows the line number of the text file
- Uses a very fast search algorithm

LFIND is written in Turbo Pascal v. 3.0 and the IBM Macro Assembler v. 2.0. The search algorithm is efficient and fast. When looking for a single word, LFIND is almost as fast as the DOS FIND command. But LFIND really comes into its own when looking for multiple words.

The search algorithm has a remarkable property: the search time is independent of the number of words sought. That is, the search takes the same amount of time whether the list contains one word, 10 words, or 50 words. The

```
program List_find ;
{ A fast, Multi-word Search Program }

  ... Excerpts of the code are included here.
  ... Access the COMPUTER LANGUAGE Bulletin Board Service or
CompuServe forum to obtain a complete listings.

var
  S_table = array[0..250, 0..63] of char;     { contains the state diagram }
  Words : array[1..150] of W_string ;                    { table of words }

{ Two assembly language routines: }
procedure X_STR(...);   external ;
procedure SEARCH(...);  external ;

procedure Initialize;
procedure Read_words;
procedure Sort_words;
procedure Open_text;
procedure Read_text;

procedure Gen_tables;
    { uses Words[] list to generate the S_table state table }
  var
    Table : integer;
    procedure Gen(Col, Row1, Row9 : integer);
      { internal proc, recursively generate table contents }
    var
      Ch1 : char;
      Ltb, Row5 : integer;
      Any_ok : boolean;

    begin                                                { internal proc }
      Ltb := Table;
      Table := Table+1;
      Max_table := Table;
      if Table > States then
        begin
          writeln(^g' Too many words/characters...cannot continue.');
          halt;
        end;
      Row5 := Row1;
      while Row5 <= Row9 do
        begin
          Ch1 := Words[Row1][Col];
          if Words[Row1][Col+1] = char(1)
          then
            begin                                { next char is Any-Char code }
              Any_ok := true;
              Words[Row1][0] := char(Col);       { ignore all after the '?' }
            end
```

Listing 1. *(Continued on page 48.)*



FIND command is faster when searching for one word, but for two or more LFIND is much better.

Many string-search algorithms have been published in computer magazines and algorithm reference books. A few years ago I was browsing through a Bell System publication and encountered a search algorithm that processed a list of words. But I have never seen this algorithm explained in a popular magazine. (The method uses, in part, a concept discussed by Michael R. Middlemas in "Transition Analysis: From One State to Another," COMPUTER LANGUAGE, Jan. 1987, pp. 31-36.)

Thinking about this novel search technique, I realized the 8086 has an ideal instruction for implementation. Like the algorithm, this instruction gets little use. The LFIND program uses this instruction in the implementation of the algorithm to achieve very rapid execution.

State-machine search

How is it possible that a search method can be equally fast when looking for 50 words as when looking for one word? The search is implemented using a state machine that is uniquely generated from the list of search words. The more words and characters in this list, the bigger the state machine becomes.

The overall size of the state machine does not have any affect on processing speed. All states are stored in RAM, and the program can jump quickly from one state to another.

During the search only one state is occupied at time, as determined by the characters encountered in the text file. Each new character means one state change. In effect, a pointer changes value based on incoming characters. A larger number of search words merely means the pointer can assume a wider range of values.

Let's assume we want to search a text file for occurrences of the words CAR, CAT?, DOG, and DUCK. (The ? character is a wildcard that means any ensuing character is accepted.) The state diagram for these words is shown in Figure 1.

The machine is set to state 0 at the first character of a word. C changes state 0 to state 1; similarly, D changes state 0 to state 4. Then, from state 1, A causes a change to state 2, and from state 4, U causes a change to state 7. And so on. Any other character abandons the state machine until a blank (b) character is encountered, which resets the machine to state 0. This reset-after-failure could be accomplished using the state machine itself, but a scan-until-blank loop is much faster.

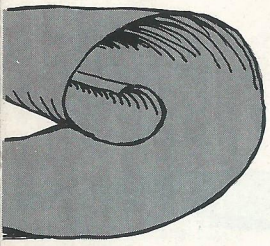
When a search word is found in the text, for example DUCKb, the state machine reaches the code for Hit that indicates a successful search. The line is then displayed on standard output along with its line number.

But this algorithm has a catch. It can only look for words, not strings. A word, as used here, is a string of alphameric characters that is preceded by white space. White space is anything that is not an alphameric character, such as a blank, tab, or carriage return character. For the state machine to work, it must be reset at the start of each word. For example, if CAT? is in the search list, we will get a hit on catalog or cat but not on bobcat.

The state diagram is useful in explaining how this search algorithm works. The next step is to implement it in a way that executes the search rapidly. The method used here trades memory for speed. A relatively large—and empty—table is

```
        else  Any_ok := false;
        repeat
          Row5 := Row5+1;
        until (Words[Row5][Col] <> Ch1)
        or (Row5 > Row9);
        Row5 := Row5-1;          { now points to 'bottom' of same-chr colm }
        if (Col > length(Words[Row1])) then
          begin
            if Row5 < Row9
              then  S_table[Ltb, Ø] := char(Bl_end)
              else  S_table[Ltb, ord(Ch1)] := char(Bl_next);
          end
        else
          if ((Row1 = Row5)
          and (Col = length(Words[Row1])))
          or (Col > length(Words[Row1]))
          or (Any_ok)
          then                                          { at End Of Word }
            begin
              if Any_ok
                then  S_table[Ltb, ord(Ch1)] := char(Any_next)
                else  S_table[Ltb, ord(Ch1)] := char(252);
            end
          else                                                { not EOW }
            begin
              S_table[Ltb, ord(Ch1)] := char(Table);
              Gen(Col+1, Row1, Row5);
            end;
        Row1 := Row5+1;
        Row5 := Row1;
      end;
  end;                                              { Gen  internal proc}
begin                                                 { main Gen_table }
  Table := Ø;
  Gen(1, 1, Wrd_count);          { start in Colm 1, Row 1 thru Wrd_count }
end;

begin                        {   M A I N   }
  Initialize;                              { set up tables and other items }
  Read_words;                          { read in the file of search words }
  Sort_words;                              { convert and sort the words }
  Gen_tables;                    { generate the state table for the search }
  Read_text;                { read in the text file and perform the search }
end.
```

Listing 1. *(Continued from page 47.)*



used to represent the state diagram. The table uses 64 bytes to represent each state; a maximum of 250 states are allowed.

By converting the state diagram in Figure 1 to a table, we get Figure 2. Each state, shown as a circled number in Figure 1, occupies a *row* in Figure 2. The state numbers are the same as the row numbers of the state table. The column heading represents an incoming character from a text file. Finally, the value in a cell is a pointer to the next valid state (row) or is a code indicating success or failure.

To use the table, always begin in row 0 with the first character of the word. For example, assume DOGb is in the text file. The first character is D. The cell at row 0, column D contains 4 as its value. This means the state changes from 0 to 4. In row 4, column 0, O moves the pointer to row 5. Continuing, G moves the pointer to row 6 and b moves it to code H, which means Hit. The line is then displayed.

The empty cells in the table actually contain a code (254) that indicates a no-match condition. When this code is encountered, a scan-until-blank loop moves a pointer to the start of the next word. Suppose the first character of the next word is T. The cell for row 0, column T, is empty because none of the words in the list start with T. The high-speed scan causes the remaining characters of this word to be ignored; the scan then readies itself for processing of the next word by the state machine. By tracing through with various words, one can see that the diagram in Figure 1 is equivalent to the table in Figure 2.

XLAT

The 8086 instruction *XLAT* (translate AL) furnishes a fast way to implement state changes. *XLAT* requires that the DS:BX registers point to the row of the table appropriate to the current state. The AL register must contain the next character. *XLAT* translates AL to the contents of the cell according to the contents of the row.

For example, set DS:BX to point to row 0 and let AL contain the code for D. After executing *XLAT,* AL is translated from D to the value 4, changing the state

State diagram for CAR, CAT?, DOG, and DUCK

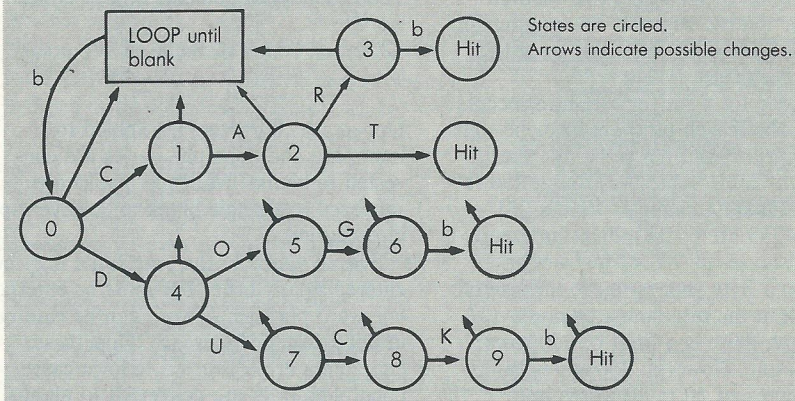


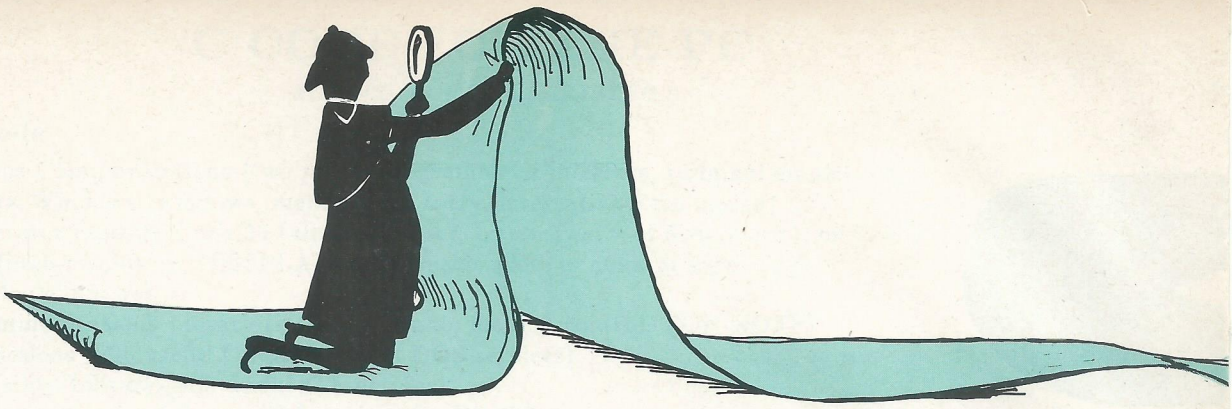
Figure 1.

State table

	Next Character read in															
	b	A	B	C	D	E	F	G	...	O	...	R	S	T	U	...
0				1	4											
1		2														
2												3		H		
3	H															
4										5					7	
5								6								
6	H															
7				8												
... 250																

Rows are states, columns are characters. Arrow shows state transitions for "DOGb."

Figure 2.



to 4. Because each row in the table is 64 bytes wide, the value 4 in the AL register is multiplied by 64 to get the updated value for the BX register. The pointer for state 4 is 256 bytes (4 × 64) from the base of the table.

If a code for a hit (or a failure) occurs instead, the search program branches out to appropriate routines to handle the condition. Table values 0 through 250 refer to valid state changes. Values 251 through 255 are codes indicating exceptions. Because the *XLAT* instruction works on a byte (not 16 bits), the size of each cell in the table must be one byte. Therefore, there is a limit of 250 rows for the table. However, 250 states can easily represent 50 to 80 words in a search list, enough for most uses.

How to use LFIND

By keying LFIND at the DOS command line, the program will display a prompt for the file names of a file containing the search-word list and the name of a text file. Or, one or both of these names can be entered from the command line. For example:

```
>lfind words.txt textfile.txt
```

means use the words found in WORDS.TXT and search the contents of TEXTFILE.TXT for any occurrences.

When a line in TEXTFILE.TXT is found that contains one or more words from the word list in WORDS.TXT, the line is displayed along with its line number. As an example, if "contains" is a search word and the text file contains this paragraph, the display would show:

```
1234:found that contains one or more
words
1236:ber. As an example, if "contains"
is a
```

Uppercase characters are folded to lowercase and nonalphamerics are converted to blanks. Multiple blanks are squeezed to a single blank to achieve rapid execution.

Search words in the search file may be entered in any order but must be entered one word per line. A word can be entered in uppercase or lowercase. Uppercase characters are converted to lowercase; nonalphamerics are converted to blanks. If more than one word appears on a line, only the first is used. For example, the file:

```
Cat?
dogs&frogs
don't
(128K or more)
The first word is accepted.
```

would be interpreted as:

```
cat?
dogs
don
128k
the
```

The program would search for only these words. (Normally, you would enter only one word per line.)

The text file can be any file containing ASCII characters. The only limitation is that each line must end with a CR character (hex 0D). A line from the text file is converted to lowercase and nonalphameric characters removed.

A word is defined as a sequence of alphameric characters enclosed in blanks. If DOG is a search word, we would get a hit on " dog " or " Dog " but not on "dogs" or "doggerel" (the leading and trailing blanks are important). If you want both DOG and DOGS, both must be in the search list. Alternatively, the ? wildcard character (as in DOG?) will yield a hit on any character string that has dog as the first three characters. Since about 80 words can be included in a search file, you have flexibility and control over the search pattern.

Generating the state table

The state table shown in Figure 2 is the key to the rapid execution of the search algorithm. The table is generated by converting to lowercase and then sorting the contents of the search file. Any duplicate words in the sorted list are removed to avoid wasting state-table entries. Duplicates would not slow the search process but would consume some of the allowed maximum of 250 table rows.

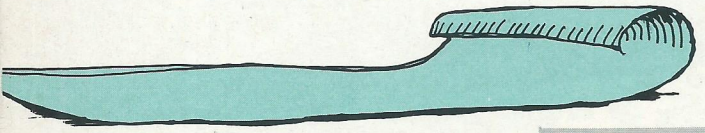
The table is generated from the sorted word list by the Pascal procedure *Gen_tables*. The sorted list is converted starting at the upper left-hand position as shown in Figure 3, where a period (.) means that this state is already represented in the table. Row 0 contains entries for c and d. The entry for c is a pointer to row 1, which represents either the a in car or cat?. The entry for d points to row 4, as shown in Figure 2.

The algorithm expressed in *Gen_tables* is presented here for the first time. It accepts as input a sorted word list and generates a state table. The internal procedure *Gen* looks down the sorted list (shown in the second column of Figure 3) until it comes to a character different from the one above it. Then it calls itself recursively, specifying the column and row(s) that, at the current level, contain the same letter. The recursive calls eventually encounter the end of a word, whereupon the procedure ends and control returns to the caller. Each encounter

Sorted word list

Search file	Sorted and folded	Processed by Gen
Dog	car	car
car	cat?	..t?
duck	dog	dog
CAT?	duck	.uck

Figure 3.



with a new (different) letter in a column adds a row to the state table.

In effect, a tree structure is traversed, wherein words that begin with the same sequence of letters are treated as sprouting from a common branch. This procedure illustrates the power of recursive processing. Without it, it would be very difficult to duplicate the action of this procedure. A much larger routine would have to be written.

X_str assembler procedure

The *X_str* procedure is called from Pascal to convert text characters to an internal code. *X_str* is called prior to sorting the word list. The search file and the text file must be converted using the same translation table the search uses so the search will be valid. Conversion moves the representation of characters to values below 64. The state table has rows that are 64 bytes wide and a translated value that is in fact an index into these rows, as discussed previously. Thus, the translation not only folds uppercase to lowercase, it also gets rid of all non-alphameric characters and at the same time moves the representation of these characters to a range below 64.

All three actions are accomplished with *XLAT*. Every character in the text file must be translated and this method is far more efficient than a complex arrangement of *IF-THEN-ELSE* statements.

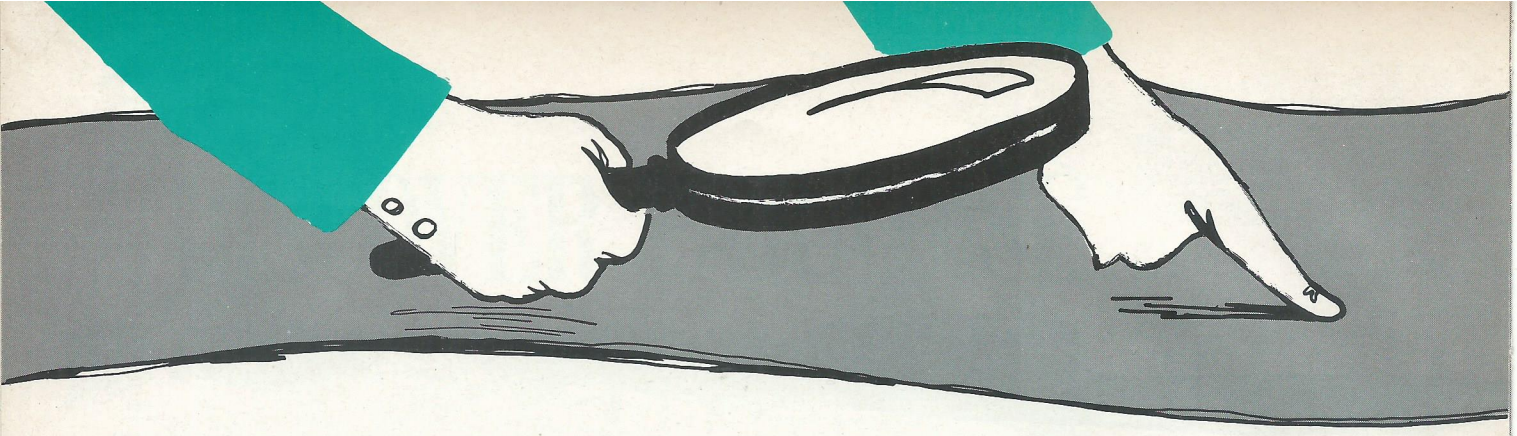
The translation table is created by the Pascal procedure *Initialize* and is stored in the *X_table* array. The *XLAT* instruction is used by the *X_str* procedure to perform the conversion. The 256 possible characters are converted to bytes ranging in value from 0 to 37. For example, the blank character is translated to 0 and the ASCII digits 0 to 9 are translated to the numbers 2 through 11.

Get_line and search

The *Get_line* and *Search* procedures shown in Listing 2 are written in assembler language for fast performance and to take advantage of specially useful 8086 machine instructions. The Pascal program reads a 12K block of bytes from the text file into a buffer and passes control to *Get_line*. *Get_line* translates one line of text at a time and saves it in a

```
; ------------------  SEARCH.ASM  -------------------------

;   The following constants are the same as in the Pascal code:
bl_end    equ   251 ; { code: means this is blank and successful search }
bl_next   equ   252 ; { code: means if blank char is next, then Ok, match }
any_next  equ   253 ; { code: means any char next is Ok, i.e., match found }
no_match  equ   254 ; { code: means no match found }
cr_code   equ   255 ;  used to indicate a carriage return code

GET_LINE        PROC NEAR
init_si :                       ; begin translation
        mov     si,[bp+24]      ; DS:SI points to Buffer contents
next:   mov     al,[si]         ; get a character from the Buffer
        cmp     al,26           ; Ctrl-Z encountered ?
        jne     no_z
; process EOF
        jmp     done            ; go home
no_z:   cmp     al,cr           ; is this a Cr character?
        jne     no_cr
;process Cr encountered in the input buffer
        cmp     ah,blnk_sp      ; is the last char a blank?
        je      bl_ok                   ; no, don't insert one, else:
        mov     byte ptr [di],blnk_sp  ;  need blank at the end string
        inc     di
bl_ok:  mov     word ptr [di],crlf      ; need this for displayed output
        inc     di
        inc     di
        mov     byte ptr [di],'$'       ;  '$' is end-of-string for DOS
        CALL    SEARCH          ; is there 'hit' in the string?
        jmp     again
no_cr:    ; process the character in AL
        xlat                    ; translate per X_table at DS:BX
        cmp     al,blnk_sp      ; is it a "blank" after translation?
        jne     no_blank        ; no, move it into T_word area
        cmp     ah,al           ; do we have 2 blanks in a row?
        je      again           ; if yes, don't put it in the string
no_blank:
        mov     ah,al           ; previous char = current char
        mov     [di],al         ; put translated character into T_word
        inc     di              ; place for next char
        cmp     di,dx           ; only enough room for 250 chrs
        jge     done            ; pretend end of line if too many
again:  inc     si              ; get next character in the Buffer
        loop    next            ; while not end of buffer
; buffer has been fully processed
        mov     si,[bp+8]       ; last DI address
        mov     [si],di         ;  must be saved for next time
done:   pop     es
        pop     ds
        pop     bp
        ret     24              ; 24  =  6 vars  *  4 bytes per var
GET_LINE        ENDP

SEARCH  PROC    NEAR
; determine if one of the search words is in the string
        mov     di,[bp+20]      ; set DI to start of the string
        inc     word ptr [di]   ; Line_counter++
        push    bx              ; save what we use
        push    cx
        push    dx
        push    di
; initialize for a dance through the state table
        mov     bx,[bp+16]      ; DS:BX points to S_table's first byte
        mov     dx,bx           ; BX saved in DX, BX reset for each word
        add     di,6            ; DI points one byte to left of string start
```

Listing 2. *(Continued on page 52.)*



```
get_char: ; get the coded next char from the next byte pos in the string
        inc     di              ; next byte
        mov     al,[di]         ; AL has it
        cmp     al,cr_code      ; is this end-of-string?
        jne     skip1           ;  no, continue
        jmp     no_hit          ;  yes, exit string search
skip1:  XLAT                    ; AL = translation per DS:BX[al]
        cmp     al,no_match     ; match failed for this word?
        je      eat_chars       ;  yes, eat nonblank chars
        cmp     al,any_next     ; any char next Ok, thus success
        je      success         ;  yes, display the string
        cmp     al,bl_next      ; blank-next state?
        je      blank_next      ;  yes, test the next char
        cmp     al,bl_end       ; blank and success?
        je      success         ;  yes, display the string
next_state: ; BX = (64 * AL) + DX :   move XLAT pointer to next state
        xor     ah,ah           ; zero out AH
        mov     cx,6            ; 2**6 = 64
        shl     ax,cl           ; AX = AX * 64
        mov     bx,dx           ; BX points to base of table
        add     bx,ax           ; BX points to base[64*AL]
        jmp     get_char        ; keep lookin'
eat_chars:  ; failure in the middle of a word, advance DI to next blank
        inc     di              ; eat nonblank
        cld                     ; look to the right for 'blank'
        mov     cx,250          ; max length of string$
        xor     al,al           ; scan for 00h = 'blank' char
        REPNE   SCASB           ;  starting at ES:DI
        mov     bx,dx           ; found it: reset for a new try
        dec     di
        jmp     get_char
blank_next:  ; if a blank is next: success; else eat chars
        inc     di
        mov     al,[di]
        cmp     al,0            ; 00h is the code for blank char
        jne     eat_chars       ; failed, eat nonblanks, try again
success:     ; convert to ASCII and display the string$
        les     di,[bp+20]      ; set ES:DI to start of T_word
        mov     ax,[di]         ; insigned integer value, line count
        xor     dx,dx           ; set up for 32-bit divide
        mov     cx,100          ; (DX:AX) / 100
        div     cx              ; quotient in DX,  rem in AX
        aam                     ; magic: AX  ==> BCD
        add     ax,3030h        ;        BCD ==> ASCII chars
        xchg    al,ah           ; reversed, per Intel way
        mov     [di+2],ax       ; lower two ASCII chars into the string
        mov     ax,dx           ; process the higher order digits
        aam
        xchg    al,ah
        add     ax,3030h
        mov     [di+4],ax
; remove leading 0's, e.g., 0031 ==> bb31
        mov     dx,di           ; use DX as end value
        mov     cx,di           ; save for later
        add     dx,4            ; last '0' is 4 bytes over
        add     di,2            ; [DI] is first '0'
zeros:  cmp     byte ptr [di],'0'
        jne     do_out          ; most significant digit
        mov     byte ptr [di],' '     ; replace '0'  with blank
        inc     di
        cmp     di,dx           ; third digit
        jbe     zeros
; convert code back to ASCII chars for display
do_out  ...
        ret
SEARCH  ENDP
```

Listing 2. *(Continued from page 51.)*

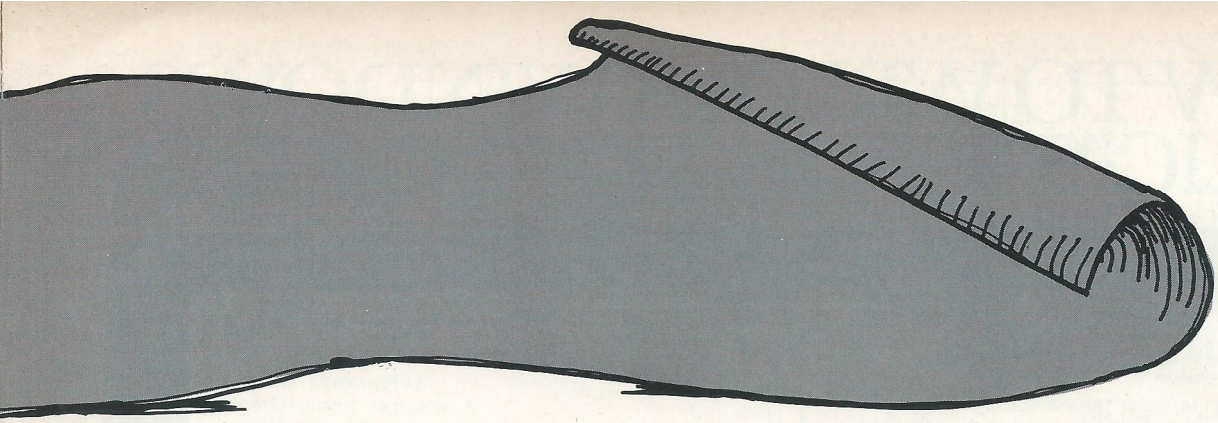
temporary buffer T_word. The line is compressed by converting all nonalphameric characters to blanks and compressing multiple blanks to one blank.

A line is defined as a sequence of bytes in the input buffer terminated by the 013 (carriage return) or 026 (EOF) characters. The translated line in T_word is then terminated with a special end-of-line byte (255) followed by a line feed and a $ character. Now it is ready to be processed by *Search* for an occurrence of a word in the list.

The strategy is first to convert an entire line and then scan it for search words. It would be possible to convert and scan at the same time, but this is not done here. The 8086 has too few registers to save the necessary pointers to the translate table, state table, input buffer, and line buffer. It is faster to set up the registers to translate a complete line of text, reset the registers, and do the scan. In addition, the search algorithm has fewer machine instructions when it works on a line that does not have multiple blanks. If a hit occurs anywhere on the line, the line is ready to be displayed without having to convert to an end-of-line. In short, it is less complicated and faster to convert a line, search a line, then unconvert and display a line if a word match occurs.

Using the state table, the *XLAT* instruction converts a translated text-file character to a value representing the next state or to a fail/success code. The state number is the next appropriate row number in the table. Since each row is 64 bytes wide, the state number is multiplied by 64 to get the offset to the next row. A shift left by six bit positions is used rather than *MUL* because shifting is much faster. Twenty machine instructions are executed per text character to walk through the state table. Only two memory accesses are performed: reading in the character and doing the *XLAT*.

If a success code is detected, the line is displayed. But more commonly a failure is detected in mid-word, so a scan-until-blank loop is executed. A *REPNE SCASB* instruction (repeat scan string, byte) is used to hunt for the next blank. In fact, most CPU time is spent in this eat_chars section of the search procedure. When *REPNE SCASB* was re-



Artwork: Dwight Been

coded to a *MOV, CMP, INC* and *LOOP* sequence, overall search times went up by a factor of almost three!

Displaying a line of text

The line number is stored in T_word as a binary integer and displayed as an ASCII character string. Conveniently, the AAM (ASCII adjust for multiply) instruction translates a hex integer in AX to a binary coded decimal (BCD) representation. The value in AX must be in the range 0 through 99 for this trick to work. Adding 3030h to AX converts BCD to ASCII decimal format. A loop replaces leading zeros with blanks.

The text buffer T_word reserves the first seven bytes of the string for the line number. The first two bytes hold a binary value representing the line number. This integer is incremented every time a string is encountered. The next four bytes are placeholders for an ASCII decimal representation of this line number. The seventh byte is :. A string looks like this:

```
<04D2h>1234:this is the line of text ... CrLf$
```

where <04D2h> is the two-byte binary line number.

As already discussed, text is internally represented as a code ranging from 0 through 37. The text is converted back to a string of ASCII characters and the 255 end-of-string byte converted to a carriage return character. After conversion the string is ready for display. DOS function 09 is used to display the text string. This function requires that the string terminate in $. The dollar sign was put there when the line was converted and compressed.

An advantage of using DOS function 09 is that output can be redirected. For example:

```
>lfind words textfile.txt > save.hit
```

directs the output to a file called SAVE.HIT.

Enlarging the search character-set

The values 39 through 63 are available to extend the character set beyond the alphamerics. You may wish to add the underscore character (_) to the list of characters looked at. Uppercase characters are folded to lowercase. Case identity can easily be preserved by modifying the translation table to include the 26 uppercase characters. However, LFIND then would become case-sensitive: "The" would be different from "the."

Remember, the search algorithm requires that a word be preceded by a blank character. So be careful about adding punctuation characters that might precede a word. If, for example, you added double quotes to the list of characters, the string "CAR" would be missed if only CAR was in the search list. You would also have to have "CAR" (with the quote character) in the search file. Avoid adding characters that often precede words.

The digits 0 through 9 (ASCII values 48 through 57) are converted to the values 2 through 11. The value 1 is assigned to the wildcard character ? and must not be reassigned. The alpha characters a through z and A through Z are translated to the values 12 through 37. All other characters are converted to 0, the code for blank. The values 38 through 63 are available to extend the search character set.

If you extend the character set, be sure to modify the *do—out* section of the assembler code. These instructions convert the 0 through 37 byte range back to ASCII characters for display.

Performance testing was done using the Pascal source file shown in Listing 1 and a 225K file containing 5,300 lines of text. A file of 80 nonsense words averaging five characters each was used to represent a worst-case situation (worst case because every line has to be completely scanned). A nonsense word also was used with LFIND, DOS FIND and WordPerfect to cause complete scanning.

The Turbo Pascal compiler is very fast, so it's included in the comparison. Of course, a code-producing compiler does more than merely scan, but it provides a comparison for those acquainted with its speed. The timings in Table 1 were run on an IBM XT.

The timings show that LFIND is about two-thirds as fast as DOS FIND. When the 80-word example is run, LFIND is more than 40 times faster than running FIND 80 times. For one word, LFIND is 3.2 times faster than a Turbo compile. If you want to find two or more words, or if case sensitivity or line numbers are factors, LFIND is superior to the DOS FIND command.

Complete versions of the source code listings are available on COMPUTER LANGUAGE's *Bulletin Board Service and CompuServe forum.*

E.J. Smith is a consultant in the San Francisco, Calif., area. He has been a programmer for 16 years and has developed PC software since 1982. His interests are algorithms, C, and personal computers.

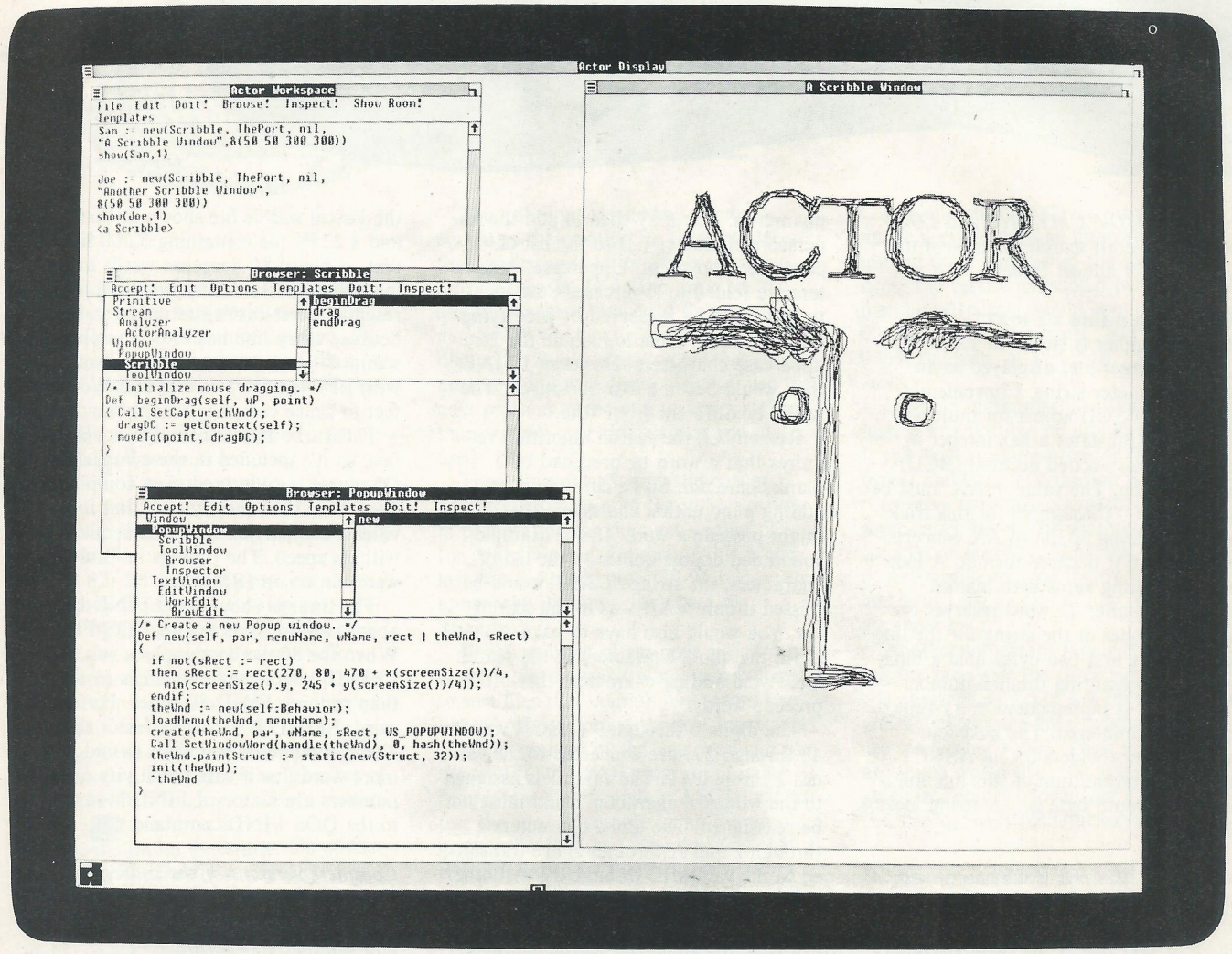
Timings

	LFIND.PAS	225K Text File
FIND, one word	0.9 sec	7.2 sec
LFIND, one word	1.2	12.5
LFIND, 80 words	1.2[1]	12.6[1]
Turbo compile	3.9	—
WordPerfect v. 4.2	4.1	93.9

[1] Not including about 1.0 sec needed to generate the state table from the 80-word search file.

Table 1.

HOW TO WRITE A WINDOWS APPLICATION IN TEN MINUTES.



Actor™ is a new language that combines Microsoft® Windows with object-oriented programming. This means you can produce mouse and window applications very quickly.

For example, we created a simple "paint" program, and used it to draw the Actor logo you see on the screen. The whole program only took ten lines and ten minutes. Part of it is in the middle window on the left.

Above, you see the commands that initialized the paint window and made it appear on the screen. Below, some code that's built into Actor, specifying window behavior. Through a process known as "inheritance," it's called into play automatically.

Try programming in this new way, and you'll never go back.

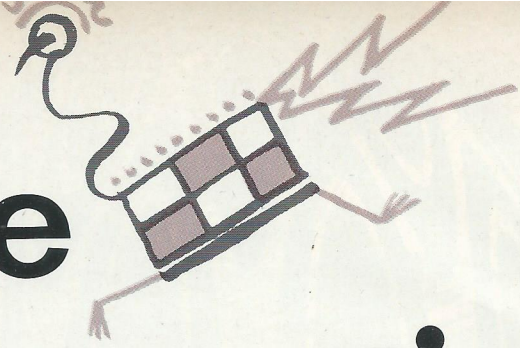
Find out about Actor. Call The Whitewater Group®, (312) 491-2370.

Technology Innovation Center
906 University Place, Evanston, IL 60201

MICROSOFT WINDOWS
AN APPLICATION DESIGNED FOR MICROSOFT WINDOWS

Now Shipping Actor 1.1 With Over 90K More Memory

CIRCLE 26 ON READER SERVICE CARD



A Recipe for Self-Decorating Eggs

By Clifford A. Pickover

High on a green hilltop in Vegreville, Canada, stands a huge model of a Ukranian Easter egg. The surface of the egg monument is covered by what appear to be symmetrical designs of colored tiles, and the reflection of the gargantuan egg in the nearby lake enhances the monument's beauty.

Many of the ornaments of modern and ancient cultures consist of symmetrical and repeating designs. Consider the beautiful Moorish motifs in tiled floors and cloths. In this article, patterns reminiscent of ornaments with a repeating motif in at least two nonparallel directions are generated. The behavior of these patterns includes a network of periodic lines and patterns and visually random fluctuations.

The idea of investigating the decorations of various cultures by considering their symmetry groups appears to have originated long ago. From a purely artistic standpoint, some of the figures produced by this article's algorithm are reminiscent of Persian carpet designs, ceramic tile mosaics, Peruvian striped fabrics, and mosque brick patterns—in addition to Moorish ornamental patterns.

Today we continue to be fascinated with symmetry and repetition in design. For modern examples, consider the isometric designs of John Locke, the geometric ornaments of Russian artist Chernikow (where simple forms create complex interweavings), and a variety of popular art deco designs. Even nature often expresses itself in terms of repeating symmetries—cross sections of plants, phase transitions, standing waves on metal plates, muscle striations, snow crystals, and dendritic ice are just a few examples.

How can the beauty of the symmetrical ornaments and designs of various cultures be simulated with the aid of a computer? The computer is a machine which, when guided by an artist, can render images of captivating power and beauty, and mathematical equations provide a vast and deep reservoir from which the artist can draw.

New recipes, such as those outlined in this article, interact with such traditional elements as form, shading, and color to produce classical and futuristic images and effects. The mathematical recipes function as the artist's helper, quickly taking care of much of the repetitive and sometimes tedious detail. By becoming familiar with computer graphics, artists with access to computers will gradually change our perception of art.

The algorithm

The egg shapes generated for this article have been called chaos eggs by some ob-

```
Pseudocode for generating patterns on the egg's surface

To aid the reader in producing the plots in this article and to
encourage future studies into equations of the reader's own
design, the following algorithm is given.  Typical parameters for
the picture boundaries and various constants are also given.

ALGORITHM Egg Tile Generator

INPUT: Res, Betal, Beta2, Gamma, Alpha, Modf

TYPICAL PARAMETER VALUES:

Betal = -11, Beta2 = -12, Gamma = 60, Alpha = 10, Res = 500, Modf = 3
Betal = -16, Beta2 = -21, Gamma = 87, Alpha =  4, Res = 500, Modf = 2
Betal = -11, Beta2 = -12, Gamma =200, Alpha =  4, Res = 500, Modf = 3

OUTPUT:  Tile patterns composed of dots at positions (i,j)

do i = 1 to res;                        (* loop in x direction          *)
  do j = 1 to res;                      (* loop in y direction          *)
   x = Betal + (Gamma*i);               (* compute x                    *)
   y = Beta2 + (Gamma*j);               (* compute y                    *)
   z = Alpha*(sin (x) + sin (y));       (* compute z                    *)
   c = trunc (z);                       (* convert z to an integer      *)
   if mod(c,modf) = 0                   (* if c evenly divisible by modf *)
      then PlotDot(i,j);                (* plot a point at position (i,j)*)
  end;                                  (* End j loop                   *)
end;                                    (* End i loop                   *)
:efig.
```

Listing 1.

Figure 1.

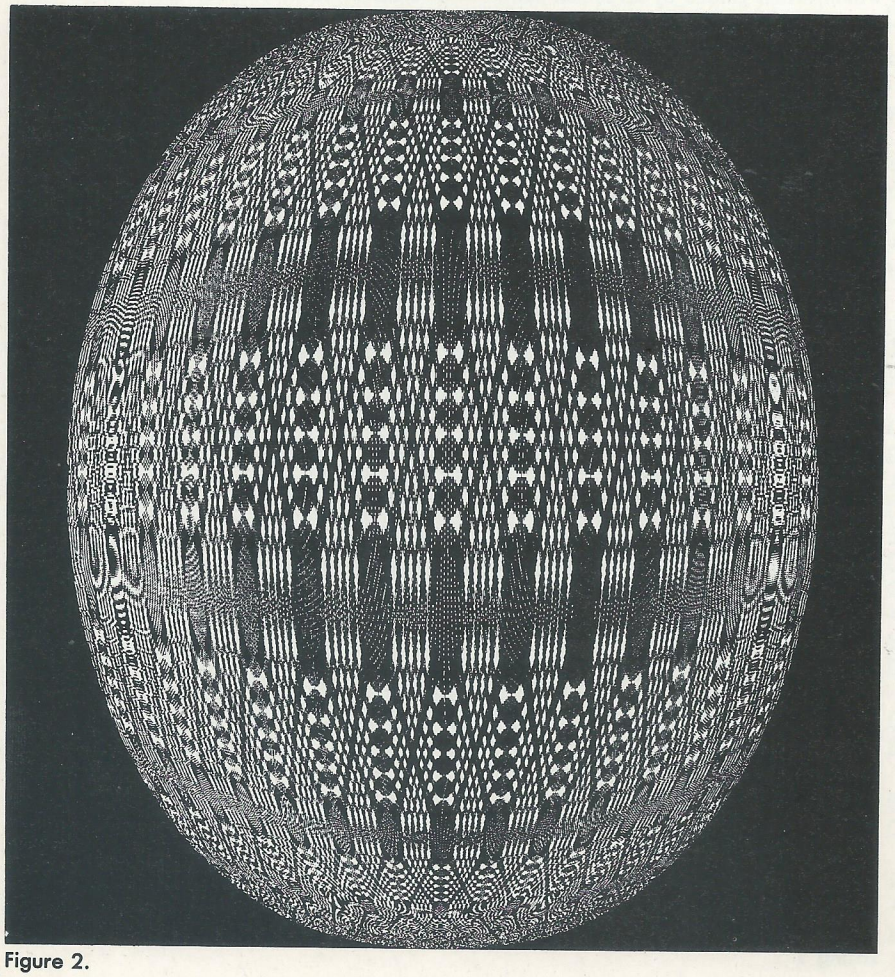
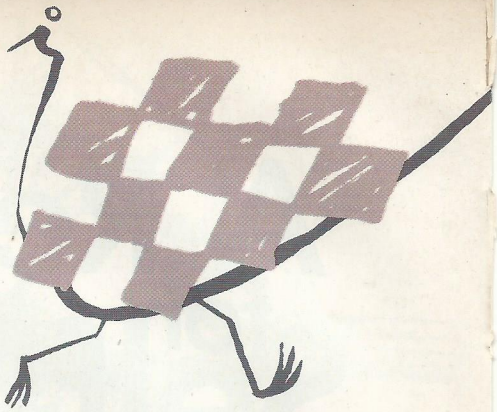


Figure 2.



servers because of their intricate designs and hard-to-predict behavior. These graphics experiments, with their variety of accompanying parameters, are good ways of characterizing the complexity of such behavior.

To compute the tiling patterns, a real number *z* is first calculated for a range of *(x,y)* pairs:

$$z_{xy} = \text{alpha } (\sin x + \sin y)$$

where alpha is a constant.

The resulting *z* value is truncated (made an integer) and divided by another integer, *m*. If the remainder is zero, a dot is plotted on the graphics screen at point *(x,y)* (Listing 1).

I first introduced this approach in "Blooming Integers" (see references). In the present article, a second transformation is applied to map this pattern to a sphere or ellipsoid. (Algorithms for spherical and other coordinate systems are not discussed here since they can be found in many computer graphics and mathematics texts. The references at the end of this article list many of these texts.)

Figures 1-3 are the resulting maps for a variety of values for alpha, *m*, and regions of *(x,y)* space. Aside from the fact that the maps are complex and aesthetically pleasing, several mathematical observations can be made.

For one, the alpha multiplier can be considered an entropy factor. As alpha increases, the disorder (entropy) of the egg increases until it looks almost random to the eye. For low values of alpha the pattern becomes perfectly ordered—somewhat crystalline in appearance. This phenomenon can be simply understood, in part, by noting that for smaller alpha the range of values for resultant *z* values is smaller, and therefore the pattern is obviously more repetitive, simple, and ordered.

Another simple observation is that as the mod factor, *m*, increases, the density of the system decreases. This too can be easily understood since fewer numbers are evenly divisible by 10 (for example) than by 3.

"No two are alike!" the saying goes when describing the intricate patterns found in snowflakes. Likewise, the few illustrations of chaos eggs here are a small set of the remarkable panoply of designs made possible with the algorithm. By turning the dial that controls *m* or alpha (or any of the other parameters in List-

designs is generated with relative computational simplicity. For this reason, the chaos eggs may be of interest to designers of museum exhibits and other educational displays for both children and adults.

Even in these simple equations, both stable and disorderly behavior can

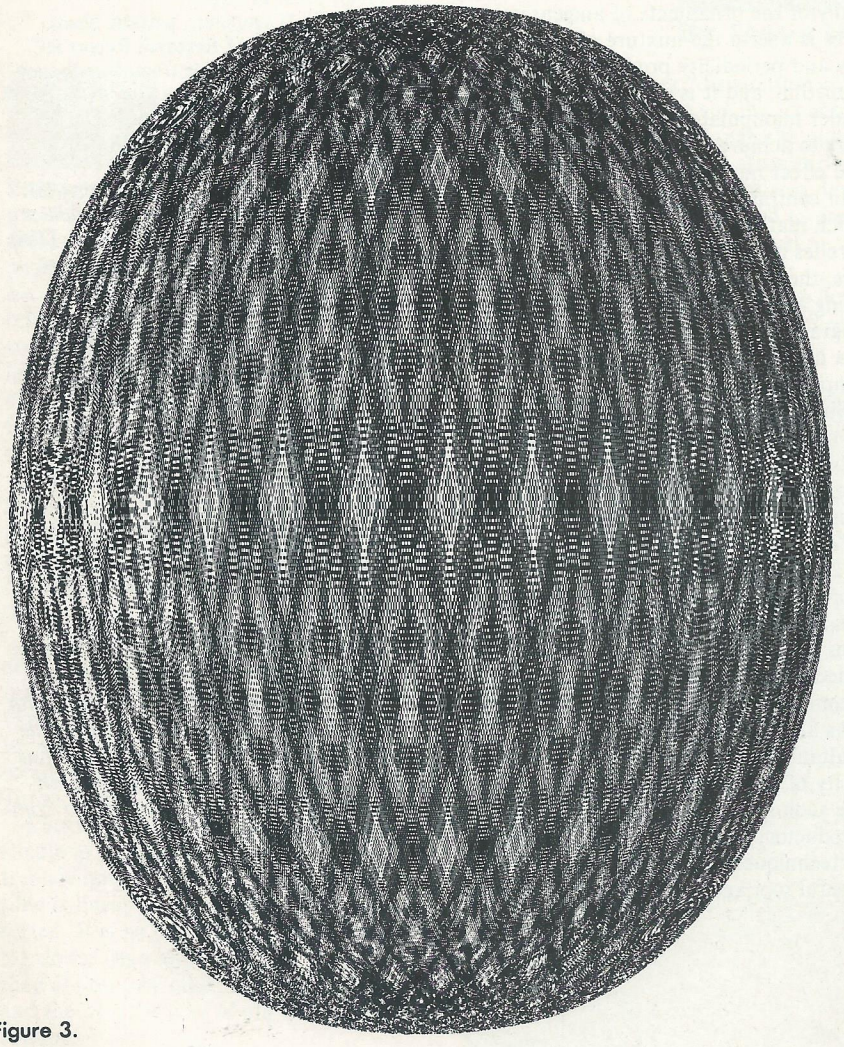


Figure 3.

Courseware Applications, Inc.

DRAWBRIDGE

Finally...a product that lets you create complex graphics displays for your applications, but saves you the tedious task of programming.

That's right...just draw the picture on the screen using the mouse or the keyboard. When you're done, Drawbridge automatically writes a program to recreate the graphic. You copy the code into your application and that's it!

Drawbridge is an interactive graphics editor that greatly increases productivity when creating sophisticated graphics displays.

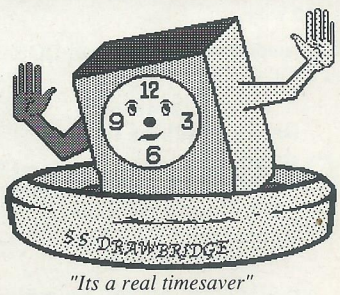
Create fantastic graphics using features such as lines, ovals, arcs, polygons, complex fill patterns, color, and text fonts.

The best way for you to evaluate this powerful new tool is to try it for yourself. The program comes with a 30-day money back guarantee.

To order, or for more information, call:

(217)359-1878

COURSEWARE APPLICATIONS, INC.
475 Devonshire Drive
Champaign, IL 61820



"Its a real timesaver"

Drawbridge is available for the IBM PC and close compatibles for $129. The current version generates C language function calls for the MetaWINDOW™ graphics library. Most popular graphics cards are supported.

CIRCLE 27 ON READER SERVICE CARD

New

An Easy To Use System For Writing Professional C Programs

Quic-PRO 5™

Fifth Generation Software Development

Quic-PRO 5™ is a complete applications development system that allows fast application development for non-professionals and pros alike.

System includes:
- Applications generator
- 4th generation language
- Query language
- Report generator
- C translator
- Datalight C compiler
- File handler
- Windowing

Fantastic Value!

Complete system..... **$199.95** (Single user)

LAN all memory..... **$595.00** models (Multi user)

Requires: PC/MS-DOS, 512 RAM, Hard Disk.

Now available in UNIX, con-DOS and muc-DOS.

30 Day Guarantee
Dealer Inquiries Invited

Q-N-E international™

136 Granite Hill Court
Langhorne, PA 19047
(215) 968-5966

VISA/MasterCard call collect for credit card order.

© 1987. Q-N-E International.
TLX 291765

CIRCLE 28 ON READER SERVICE CARD

emerge—the difference caused by a minute change in the input values. This behavior is qualitatively similar to that in chaos theory—the study of how complicated behavior can arise in systems based on simple rules and how minute changes in the input of a system can lead to large differences in the output.

The beauty and complexity of Figures 1-3 (and the many other shapes that can result from this algorithm) correspond to behavior no one could fully have appreciated or suspected before the age of the computer. This complexity makes it difficult to objectively characterize such structures. Therefore it is useful to develop graphics systems that allow the maps to be followed in a qualitative and quantitative way.

Though the equations seem to display what might be called bizarre behavior, there nevertheless seems to be a limited repertory of recurrent patterns.

The resemblance of the figures produced by the algorithm discussed here to many of the ornaments of ancient cultures is due to the mixture of stochasticity and periodicity produced by the algorithm, and it is suggested that the reader manipulate the various parameters to achieve artistic control of the visual effect most desired.

In contrast to previous systems in which mathematical and aesthetic beauty relies on the use of imaginary numbers, these calculations use real numbers. Doing so also facilitates their study with programming languages having no complex data types on small personal computers.

Since the original research in this article was done, an excellent article on a similar technique by Brian Hayes came to my attention ("On the bathtub algorithm for dot-matrix holograms," *COMPUTER LANGUAGE*, Mar. 1986, pp. 21-25). Hayes introduces the truncation approach as applied to the parabolic equation $z = x^2 + y^2$. His planar patterns are different than the figures presented here, and the reader is encouraged to compare the figures in both articles.

As already mentioned, the richness of resultant forms contrasts with the simplicity of the generating formula. An article such as this can only be viewed as introductory; however, it is hoped that the techniques and equations will provide a useful tool and stimulate future studies in the graphic characterization of the morphologically rich tiling patterns produced by such a simple generating formula.

I owe a special debt of gratitude to Alan Hauck (IBM) for familiarizing me with algorithms that map drawings to spherical surfaces.

References

Audsley, W. and G. *Designs and Patterns from Historic Ornament*. New York: Dover, 1968.

Dewdney, A. "Computer Recreations," *Scientific American*, Sept. 1986: 14-23.

Dowlatshahi, A. *Persian Designs and Motifs*. New York: Dover, 1979.

Hayes, B. "On the bathtub algorithm for dot-matrix holograms," *COMPUTER LANGUAGE*, Mar. 1986: 21-25.

Pickover, C. "Mathematics and Beauty I: time-discrete phase planes associated with the cyclic system, $(x(t) = -f(y(t)), y(t) = f(x(t)))$, *Computers and Graphics*, 11(2), 1986: 217-226.

Pickover, C. "Computers, pattern, chaos, and beauty." IBM Research Report RC 12281, 1986. Available from: Distribution Services, IBM, Yorktown Hts., N.Y. 10598.

Pickover, C. "Graphics, bifurcation, order and chaos," *Computer Graphics Forum*, June 1987: 26-33.

Pickover, C. "Mathematics and Beauty III: Blooming Integers," *Computer Graphics World* (March, 1987), 10(3): 54-57. [This article describes my first planar studies with truncated sinusoids and provides additional graphics.]

Pickover, C. "On the educational uses of computer-generated cartoon faces," *Journal of Educational Technology Systems*, 13: 185-198.

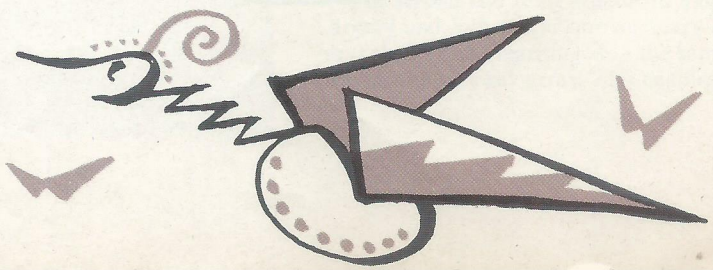
Peterson, I. "Picture This," *Science News* 131(25): 392-395.

Rowe, W. *Flora and Fauna Design Fantasies*. New York: Dover, 1976.

Speigel, M. *Schaum's Outline Series: Mathematical Handbook*. New York: McGraw-Hill, 1968.

Clifford Pickover is a research staff member at the IBM Thomas J. Watson Research Center, Yorktown Heights, N.Y. He received his Ph.D. in molecular biophysics and biochemistry from Yale University, New Haven, Conn., and has since published numerous articles on novel graphical representations for complicated data.

Artwork: Steve Campbell



Upgrade your technology

The software technology available to programmers of IBMcompatible personal computers is truly amazing. And newer, more powerful development packages appear all the time. But until now, finding out about these important products has been a difficult and time consuming task.

FREE Buyer's Guide. The New 76 page Programmer's Connection Fall 1987 Buyers Guide contains individual descriptions of over 500 titles of programmer's development software by over 150 manufacturers. Each description covers major product features as well as any software or hardware requirements and version numbers. In the box on the right are some examples of the types of descriptions you'll find in our Buyer's Guide.

No Hidden Charges. The low discount prices in our Buyer's Guide are all you pay. We don't charge extra for UPS Ground shipping, credit cards, COD orders, purchase orders, sales tax (except Ohio) or special handling (except for non-Canadian international orders).

Guarantees. We offer FREE 30-day no-risk return guarantees and 30-day evaluation periods on most of our products.

Latest Versions. The products we carry are the latest versions and come with the same manufacturer's technical support as if buying direct.

Large Inventory. We have one of the largest inventories of programmer's development products in the industry. Most orders are shipped within 24 hours.

Noncommissioned Staff. Our courteous salespeople are always ready to help you. And if you aren't sure about your needs, our knowledgeable technical people can give you sound, objective advice.

Experience. We've specialized in development software for IBMcompatible personal computers since 1984 and are experienced at providing a full range of quality products and customer services.

How to Get Your Copy. There are three ways for you to receive your FREE copy of the Programmer's Connection Buyer's Guide: 1) Use the reader service card provided by this journal; 2) Mail us a card or letter with your name and address; or 3) Call one of our convenient toll free telephone numbers.

If you haven't yet received your Programmer's Connection Fall 1987 Buyer's Guide, act now. Upgrading your programming technology could be one of the wisest and most profitable decisions you'll ever make.

Blaise
C TOOLS PLUS/5.0
List $129 Ours $99

C TOOLS PLUS/5.0 is a library for Microsoft C that can provide you with a full spectrum of general-purpose utility functions. Included are functions that take advantage of the machine features of IBM-compatible personal computers and DOS and complement the standard compiler library. Almost all functions are written in C using techniques most suitable for good C program design. Some of the areas covered are: extensive string handling; screen handling including support for multiple monitors and the EGA; general utility and keyboard functions; DOS memory management; windows that can be stacked, removed and accept user input; intervention code; and interrupt service routine support for truly flexible resident applications. C TOOLS PLUS/5.0 includes all source code, complete examples and a comprehensive reference manual.

Supports Microsoft C 5.0 and Microsoft QuickC.

ESP
Command Plus
List $80 Ours $69

Command Plus is a powerful, bootable MS-DOS command processor fully compatible with COMMAND.COM. Many new features are included that will help you increase your speed, productivity and ease of programming. These include a history processor that lets you recall and edit previously entered commands using the cursor keys. The alias facility allows the creation of command macros with replaceable parameters. Other features include: regular expressions in filenames; directory and argument stacks; a command line editor; access to environment variables; a MOVE command; and Browse, a full screen file viewer. Command Plus also includes SCRIPT, a batch processor that uses a Pascal-like language that features: integer and string variables; boolean, math and string operators; CALL, FOR/WHILE, GOTO, IF/THEN/ELSE, and SWITCH statements; and display and file access routines. Enhancements to DIR include sort options and file attrib display. COPY options include selection by date/time range, and recursive sub-directory processing.

Requires 48K memory. Version 1.2.

Meridian Software
AdaVantage Compiler 2.0
List $795 Ours $735

The Meridian AdaVantage Compiler is a fully validated implementation of the Ada language. The compiler generates native 8086 code in the Intel standard object format. A linker is provided to create stand-alone executable files for MS-DOS. The Meridian AdaVantage Compiler package includes the compiler, linker, library management tools, support packages, runtime libraries and a configuration tool. In addition to the standard Ada packages, support packages are provided to make integer, floating point and text I/O more convenient to implement. The compiler also provides a pragma INTERFACE used to make calls to subprograms written in 8086 assembly language, Meridian Cor Meridian Pascal. A source level debugger will be available in late 1987.

Requires hard disk and 640K memory. Runtime fees apply if more than 99 copies are sold. Version 2.0.

CALL TOLL FREE

USA: 800-336-1166
Canada: ... 800-225-1166
Ohio & Alaska (Collect): . . 216-494-3781
International: 216-494-3781
Telex: 9102406879
Easylink: 62806530

Programmer's Connection
7479 Whipple Ave. NW
North Canton, OH 44720

Please turn the page for our latest price list and ordering information.

PROGRAMMER'S QUALITY INTEGRITY Service SUPPORT PRICE CONNECTION

The Free Programmer's Connection Fall '87 Buyer's Guide.

CIRCLE 29 ON READER SERVICE CARD

ai - expert systems

	List	Ours
1st-CLASS *by Programs in Motion*	495	399
EXSYS *Development Software by EXSYS*	395	309
EXSYS Runtime System	600	469
LEVEL5 *by Information Builders* ... *New*	685	569
Logic-Line Series *All varieties by Thunderstone*	CALL	CALL

ai - lisp language

Golden Common LISP *by Gold Hill*	495	CALL
Golden Common LISP Developer *by Gold Hill*	1190	CALL
Q'Nial *Various by NIAL Systems*	CALL	CALL
Star Sapphire LISP *Compiler by Sapiens* ... *New*	495	429
TransLISP PLUS *from Solution Systems*	195	125

ai - prolog language

Arity Combination Package	1095	979
Expert System Development Pkg	295	229
File Interchange Toolkit	50	44
PROLOG Compiler & Interpreter	650	569
Screen Design Toolkit	50	44
SQL Development Package	295	229
Arity PROLOG Interpreter	295	229
Arity Standard Prolog	95	77
LPA microPROLOG *All Varieties*	CALL	CALL
MPROLOG Language Primer *LOGICWARE*	50	45
MPROLOG P500 *by LOGICWARE*	495	395
MPROLOG P550 *w/Primer by LOGICWARE*	220	175
Turbo PROLOG *by Borland Intl*	100	64
Turbo PROLOG Toolbox *by Borland Intl*	100	64

ai - smalltalk language

Smalltalk/V	100	84
EGA/VGA Color Option	50	45
Goodies Diskette	50	45
Smalltalk/Comm	50	45

ai - texas instruments

Arborist *Decision Tree Software*	595	519
PC Scheme *Lisp* ... *New Version*	95	84
Personal Consultant Easy ... *New Version*	495	435
Personal Consultant Image	495	435
Personal Consultant Online	995	869
Personal Consultant Plus ... *New Version*	2950	2589
Personal Consultant Runtime	95	84

ada language

AdaVantage *GSA-Validated by Meridian Software*	795	735
AdaVantage Utility Packages	50	47
DOS Environment Package	50	47
Janus/ADA C Pak *by R&R Software*	95	84
Janus/ADA D Pak *by R&R Software*	1250	1059
Janus/ADA ED Pak *by R&R Software*	395	349

apl language

APL*PLUS PC *by STSC*	695	495
APL*PLUS PC Spreadsheet Mgr *by STSC*	195	139
APL*PLUS PC Tools Vol 1 *by STSC*	295	199
APL*PLUS PC Tools Vol 2 *by STSC*	85	58
APL*PLUS PS/2 *by STSC* ... *New*	695	495
ATLAS*GRAPHICS *by STSC*	450	329
Financial/Statistical Library *by STSC*	275	189
Pocket APL *by STSC*	95	69
STATGRAPHICS *by STSC*	895	649

assembly language

386 ASM/LINK *Cross Asm by Phar Lap*	495	389
8088 Assembler *w/Z-80 Translator by 2500 AD*	100	89
ASMLIB *Function Library by BCSoft*	149	125
asmTREE *B-Tree Dev System by BCSoft*	395	329
Cross Assemblers *Various by 2500 AD*	CALL	CALL
EZASM *by C Source*	70	59
Microsoft Macro Assembler	150	93
Turbo Debugger *by Speedware*	89	79
Turbo Editasm *by Speedware*	99	84
Visible Computer: 8088 *by Software Masters*	80	64

basic language

db/Lib *for QuickBASIC by AJS Publishing*	139	119
Finally *by Komputerwerk*	99	85
MACH 2 *by Micro Help*	69	55
Microsoft QuickBASIC	99	63
QBase *Relational Database by Crescent*	89	79
Quick-Tools *by BCSoft*	130	109
QuickPak *by Crescent Software*	69	59
Scientific Subroutine Library *by Peerless*	125	99
Screen Sculptor *by Software Bottling*	125	91
Stay-Res *by MicroHelp*	69	55
True Basic *w/Run-time* ... *Special Price*	200	99
True Basic	100	79
Run-time Module	100	79
Various Utilities	50	41
Turbo BASIC Compiler *by Borland Intl*	100	64

blaise products

ASYNCH MANAGER *Specify C or Pascal*	175	135
C TOOLS PLUS/5.0 ... *New Version*	129	99
KeyPlayer *Super Batch Program* ... *New*	50	45
LIGHT TOOLS *for Datalight C*	100	65
PASCAL TOOLS	125	95
PASCAL TOOLS 2	100	79
PASCAL TOOLS & TOOLS 2	175	135
RUNOFF *Text Formatter*	50	45
TURBO ASYNCH PLUS	100	79
TURBO C TOOLS	129	99
TURBO POWER TOOLS PLUS	100	79
VIEW MANAGER *Specify C or Pascal*	275	199

borland products

EUREKA *Equation Solver*	167	105
Reflex: The Analyst	150	99
Sidekick	85	57
Superkey	100	64
Turbo Basic Compiler	100	64
Turbo Basic Database Toolbox	100	64
Turbo Basic Editor Toolbox	100	64
Turbo Basic Telecom Toolbox	100	64
Turbo C Compiler *(Call for support products)*	100	64
Turbo Lightning and Word Wizard	150	94
Turbo Lightning	100	64
Turbo Lightning Word Wizard	70	47
Turbo Pascal and Tutor ... *New Version*	CALL	CALL
Turbo Pascal ... *New Version*	100	64
Turbo Pascal Tutor ... *New Version*	70	41
Turbo Pascal Database Toolbox ... *New Version*	100	64
Turbo Pascal Editor Toolbox ... *New Version*	100	64
Turbo Pascal Gameworks Toolbox ... *New Version*	100	64
Turbo Pascal Graphix Toolbox ... *New Version*	100	64
Turbo Pascal Numerical Methods Toolbox ... *New*	100	64
Turbo Prolog Compiler	100	64
Turbo Prolog Toolbox	100	64

c compilers

C86PLUS *by Computer Innovations*	497	359
DeSmet C *w/Debugger & Large case*	209	184
DeSmet C *w/Debugger only*	159	138
Eco-C *Complete System by Ecosoft*	140	119
Instant C *by Rational Systems*	495	369
Instant-C/16M *by Rational Systems* ... *New*	895	695
Lattice C Compiler *vers. 3.2 from Lattice*	500	265
Mark Williams Let's C w/csd	75	54
Microsoft C Compiler w/CodeView ... *New Version*	450	269
Microsoft QuickC Compiler ... *New*	99	63
Optimum-C *by Datalight*	139	95
Turbo C Compiler *by Borland*	100	64
Uniware 68000/10/20 Cross Compiler *by SDS*	995	829

c interpreters

C-terp *by Gimpel, Specify compiler*	298	219
C Trainer *with Book by Catalytix*	122	87
Introducing C *by Computer Innovations*	125	99
Run/C *by Age of Reason*	120	69
Run/C Professional *by Age of Reason*	250	145

c utilities

Blackstar C Library *by Sterling Castle* ... *New*	125	98
C++ *by Guidelines w/version 1.1 kernel*	195	172
c-tree & r-tree Combo *by FairCom*	650	519
c-tree *ISAM File Manager*	395	315
r-tree *Report Generator*	295	239
Csharp Realtime Toolkit *by Systems Guild*	600	489
Curses Window Dev Pkg *by Aspen Scientific* ... *New*	119	105
with Source Code	289	249
dBx *dBASE to C Translator by Desktop AI*	350	299
with Source Code	550	419
Flash-up Windows *by Software Bottling*	90	78
GraphiC *Color version by Sci Endeavors*	350	274
GRAFLIB *by Sutrasoft*	175	159
HALO *Graphics by Media Cybernetics*	300	205
HALO Development Pkg *for Microsoft*	595	389
The HAMMER *by OES Systems*	195	129
PANEL *Forms Management by Roundhill*	295	215
PANEL/TC *for Turbo C by Roundhill*	129	95
PANEL Plus *by Roundhill*	495	395
PC Lint *by Gimpel Software*	139	99
PLOTHP *by Sutrasoft*	175	159
RTC PLUS *Fortran to C by Cobalt Blue*	450	399
Sapiens V8 *Virtual Memory Manager* ... *New*	300	265
Scientific Subroutine Library *by Peerless*	175	135
TE Text Editor *source by Sub Systems*	95	85
Vitamin C *by Creative Programming*	225	149
VC Screen *Forms Designer*	100	79
Zview *by Data Mgmt Consultants*	245	139

cobol language

COBOLspII *by Flexus*	395	329
FPLIB *for Realia COBOL by BCSoft*	149	129
Micro Focus COBOL *See Micro Focus Section*		
Microsoft COBOL *See Microsoft Section*		
PCDT *by Pro-Code*	995	895
Realia COBOL with RealMENU	1145	899
Realia COBOL	995	783
RealCICS	995	783
RM/COBOL *by Ryan-McFarland*	950	639
RM/COBOL 85 *by Ryan-McFarland*	1250	895
RM/NET+5 *by Ryan-McFarland* ... *New*	300	259
RM/Screens ... *New*	395	334
SCREENIO *by Norcom*	400	379
screenplay *for COBOL by Flexus*	175	129

css products

Combo Package *by Custom Software Systems*	199	175
PC/SPELL *Spelling Checker*	49	45
PC/TOOLS *UNIX-like Utilities*	49	45
PC/VI *vi Editor*	149	99

debuggers & profilers

386 DEBUG *Cross Debugger by Phar Lap*	195	129
Advanced Trace-86 *by Morgan Computing*	175	115
Codesmith-86 *by Visual Age*	145	98
DSD87 *by Soft Advances*	125	79
MiniProbe *by Atron*	395	369
Periscope I *with Board by Periscope*	345	275
Periscope II *with NMI Breakout Switch*	175	139
Periscope II-X *Software only*	145	105
Periscope III *8 MHz version*	995	795
Periscope III *10 MHz version*	1095	875
The PROFILER *with Source Code by DWB*	125	89
TURBOsmith *Source debugger for Turbo Pascal*	99	89
The WATCHER *Profiler by Stony Brook*	60	51

disk utilities

Back-It *by Gazelle Systems* ... *New Version*	130	115
Disk Optimizer *by Softlogic Systems*	60	55
Disk Technician *by Prime Solutions* ... *New*	100	89
FASTBACK *by 5th Generation Systems*	179	129
Vcache *by Golden Bow Systems*	50	47
Vopt *by Golden Bow Systems*	50	47
Vfeature *by Golden Bow Systems*	80	74
Vfeature Deluxe *by Golden Bow Systems*	120	111
XenoCopy-PC *by XenoSoft*	80	69

dos utilities

Advanced Norton Utilities	150	99
Command Plus *by ESP Software*	80	69
Desqview *from Quarterdeck*	100	85
FANSI-CONSOLE *by Hersey Micro*	75	62
Mace Utilities *Paul Mace Software* ... *New*	99	89
MicroHelp Utility *by MicroHelp*	59	49
Norton Commander *by Peter Norton*	75	55
Norton Utilities *by Peter Norton*	100	59
OPAL *Shell Language by Software Factory*	99	89
Q-DOS II *by Gazelle Systems*	70	59
Taskview *by Sunny Hill Software*	80	55

essential products

C Utility Library	185	119
Essential Comm Library *with Debugger*	250	189
Essential Comm Library *Software Only*	185	125
Breakout Debugger Only *Any language*	125	89
Essential Graphics	250	183

forth language

CFORTH *Native Code Compiler by LMI*	300	229
FORTH/83 Metacompiler *Specify Target*	750	599
PC/FORTH *by Laboratory Microsystems*	150	109
PC/FORTH+ *by Laboratory Microsystems*	250	199
Programmer's Package #1 *by LMI* ... *New*	250	199
Programmer's Package #2 *by LMI* ... *New*	350	279
Programmer's Package #3 *by LMI* ... *New*	500	399
UR/FORTH *Also Available for OS/2 by LMI*	350	279
UR/FORTH Libraries	500	395

fortran language

50 MORE: FORTRAN *by Peerless Scientific*	125	95
ACS Time Series *by Alpha Computer Service*	495	389
AUTOMATED PROGRAMMER *by KGK Automated*	995	949
Essential Graphics *by Essential Software*	250	183
Forlib-Plus *by Alpha Computer Service*	70	44
FORTLIB *by Sutrasoft*	125	109
FORTRAN Addendum *by Impulse Engr*	95	85
FORTRAN Addenda *by Impulse Engr*	165	138
GRAFLIB *by Sutrasoft*	175	159
HALO *Graphics by Media Cybernetics*	300	205
I/O PRO w/No Limit Library *by MEF*	250	219
Microcompatibles Combo Package	240	215
Grafmatic	135	117
Plotmatic	135	117
Microsoft FORTRAN *w/CodeView*	450	269
No Limit Library *by MEF Environmental*	129	109
Numerical Analyst *by MAGUS*	295	249
PANEL *by Roundhill Computer Systems*	295	215
PLOTHP *by Sutrasoft*	175	159
RM/FORTRAN *by Ryan-McFarland* ... *New Version*	599	399
RTC PLUS *Fortran to C by Cobalt Blue*	450	399
Scientific Subroutine Lib *by Peerless*	175	135
Statistician *by Alpha Computer Service*	295	235
STATLIB.GL: *by Peerless*	295	239
STATLIB.TSF: *by Peerless*	295	239
Strings & Things *by Alpha Computer Service*	70	45

greenleaf products

Greenleaf C Sampler *for Turbo C & QuickC* ... *New*	95	69
Greenleaf Comm Library	185	125
Greenleaf Data Windows Library	225	155
with Source Code	395	249
Greenleaf Functions	185	125

help utilities

HELP/Control *by MDS*	125	99
On-line Help *from Opt-Tech*	149	99
SoftScreen/HELP *by Dialectic Systems*	195	149

lattice products

Lattice C Compiler *ver 3.2 from Lattice*	500	265
with Library Source Code	900	495
C Cross Reference Generator	50	37
with Source Code	200	139
C-Food Smorgasbord *Function Library*	150	95
with Source Code	300	179
C-Sprite *Source Level Debugger* ... *Special Offer*	175	CALL
Curses *Screen Manager*	125	85
with Source Code	250	169
dBC III	250	169
with Source Code	500	356
dBC III Plus	750	594
with Source Code	1500	1184
LMK *Make Facility*	195	138
RPG II Combo *All three items below*	1100	875
RPG II Compiler *No Royalties*	750	625
SEU *Source Entry Utility*	250	199
Sort/Merge	250	199
Screen Design Aid Utility *for RPG II*	350	309
SecretDisk II *Encryption Utility* ... *New*	120	88
SideTalk *Resident Communications*	120	88
SSP/PC *Scientific Subroutine Library*	350	269
Text Management Utilities	120	88

metagraphics products

LightWINDOW/C *for Datalight C*	95	79
FontWINDOW	95	79
FontWINDOW/PLUS	275	229
MetaWINDOW *No Royalties*	195	159
MetaWINDOW/PLUS	275	229
TurboWINDOW/C *for Turbo C*	95	79
TurboWINDOW/Pascal *for Turbo Pascal*	95	79

micro focus products

Micro Focus COBOL/2	900	729
Micro Focus Level II COBOL *w/Animator*	495	395
Level II COBOL	349	279
Level II Animator	195	155
Micro Focus Level II COBOL/ET for UNIX	CALL	CALL
Micro Focus PC-CICS ... *New*	1495	1189
with **Micro/SPF** ... *New*	1595	1269
Micro Focus Personal COBOL	149	119
Micro Focus Professional COBOL	2000	1595
Micro Focus VS COBOL/XENIX	1495	1195
Micro Focus Support Products:		
COBOL/IQ *Ad hoc Report Writer*	495	395
COBOL/IQ *for DOS 3.X Networks*	995	795
FORMS-2	295	235
SOURCEWRITER	995	795

microport products

386 Unlimited License Kit	249	209
AT Unlimited License Kit	249	209
DOSMerge286 *Run DOS and UNIX together*	149	129
DOSMerge386 *Run DOS and UNIX together*	CALL	CALL
System V/386 Combination	799	699
386 Runtime System	199	169
386 Software Development System	499	429
Text Preparation System	199	169
System V/AT Combination	549	465
AT Runtime System	199	169
AT Software Development System	249	209
Text Preparation System	199	169

microsoft products

Microsoft BASIC Compiler *for XENIX*	695	419
Microsoft BASIC Interpreter *for XENIX*	350	209
Microsoft C Compiler *w/CodeView ... New Version*	450	269
Microsoft COBOL Compiler *with COBOL Tools*	700	429
for XENIX	995	609
Microsoft FORTRAN *Optimizing Compiler/CodeView*	450	269
Microsoft FORTRAN *for XENIX*	695	419
Microsoft Learning DOS	50	36
Microsoft MACH 10 *with Mouse & Windows*	549	369
Microsoft MACH 10 *Board only*	399	279
Microsoft Macro Assembler ... *New Version*	150	93
Microsoft Mouse *for IBM PS/2*	CALL	CALL
Microsoft Mouse *Bus Version*	CALL	CALL
Microsoft Mouse *Serial Version*	CALL	CALL
Microsoft Pascal Compiler	300	179
for XENIX	695	419
Microsoft QuickBASIC	99	63
Microsoft QuickC ... *New*	99	63
Microsoft Windows	99	63
Microsoft Windows Development Kit	500	299
Microsoft Word ... *New Version*	450	269

mks products

MKS AWK ... *New*	75	65
MKS RCS *Revision Control System* ... *New*	189	155
MKS Toolkit *with MKS VI Editor*	139	109
MKS Trilogy *with AWK, CRYPT & Korn Shell* ... *New*	119	99
MKS VI *Editor by MKS* ... *New*	75	65

modula-2 language

LOGITECH Modula-2 Development System ... *New*	249	199
Modula-2 Compiler Pack ... *New*	99	79
Modula-2 Toolkit ... *New*	169	139
LOGITECH Modula-2 ROM Pkg	299	239
LOGITECH Modula-2 Window Pkg	49	39
Macro2 *Macro preprocessor by PMI*	89	79
ModBase *by PMI*	89	79
ModGraph *by TEQNA* ... *New*	50	45
Repertoire *by PMI*	89	75
Science & Engrg Tools *by Quinn-Curtis*	75	67
Universal Graphics Library *by Quinn-Curtis*	130	119

mouse products

LOGIMOUSE BUS *with PLUS Pkg by LOGITECH*	119	98
with PLUS & PC Paintbrush	149	119
with PLUS & CAD Software	189	153
with PLUS & CAD & Paint	219	179
with First Publisher ... *New*	CALL	CALL
LOGIMOUSE C7 *with PLUS Pkg, Specify Connector*	119	98
with PLUS & PC Paintbrush	149	119
with PLUS & CAD Software	189	153
with PLUS & CAD & Paint	219	179
with First Publisher ... *New*	CALL	CALL
Microsoft Mouse *See Microsoft Section*		

other languages

ACTOR *by Whitewater Group* ... *New Version*	495	419
CCS MUMPS *Single-User by MGlobal*	60	50
CCS MUMPS *Single-User Multi-Tasking*	150	129
CCS MUMPS *Multi-User*	450	359
Marshal Pascal *by Marshal Language Systems*	189	165
Pascal-2 *by Oregon Software*	395	325
Personal REXX *by Mansfield Software*	125	99
SNOBOL4+ *by Catspaw*	95	80

other products

Carbon Copy Plus *by Meridian Technology*	195	159
Dan Bricklin's Demo Pgm *by Software Garden*	75	57
Dan Bricklin's Demo Tutorial	50	45
Fast Forward *by Mark Williams*	70	59
Instant Replay *by Nostradamus*	150	CALL
MicroTEX *Typesetting from Addison-Wesley*	295	CALL
Printer Drivers ... *New*	CALL	CALL
Net-Tools *by BCSoft*	149	129
Norton Guides *Specify Language* ... *New*	100	75
OPT-Tech Sort *by Opt-Tech Data Proc*	149	99
PC/TOOLS *by Custom Software*	49	45
Screen Machine *by MicroHelp*	79	59
SuperSort *by LifeStyle* ... *New*	139	119

phoenix products

Pasm86 *Macro Assembler version 2.0*	195	108
Pdisk *Hard Disk & Backup Utility*	145	99
Pfantasy Pac *Phoenix Combo*	995	595
Pfinish *Execution Profiler*	395	209
Pfix86plus *Symbolic Debugger*	395	209
PforCe *Specify C Compiler*	395	209
PforCe++ *Specify C Compiler and C++*	395	209
Plink86plus *Overlay Linker*	495	275
Pmaker *Make Utility*	125	78
Pmate *Macro Text Editor*	195	108
Pre-C *Lint Utility*	295	154
Ptel *Binary File Transfer Program*	195	108

polytron products

PolyBoost *Software Accelerator* ... *Special Price*	80	54
PolyDesk III	99	72
PolyDesk III Archivist	50	42
PolyDesk III Cryptographer	50	42
PolyDesk III Talk	70	52
PolyLibrarian *Library Manager*	99	89
PolyLibrarian II *Library Manager*	149	129
PolyMake *UNIX-like Make Facility*	149	129
PolyShell ... *Special Price*	149	105
Polytron C Beautifier	50	45
PolyXREF *Complete Cross Ref Utility*	219	185
PolyXREF *One language only*	129	109
PVCS Corporate *Version Control System*	395	329
PVCS Personal	149	129

program mgmt utilities

Interactive EASYFLOW *by Haventree* .. *New Version*	150	125
PrintQ *by Software Directions*	89	84
Quilt Computing Combo Package	250	199
QMake *Program Rebuild Utility*	99	79
SRMS *Software Revision Mgmt System*	185	159
Sapiens MAKE & V8 ... *New*	439	379
Sapiens MAKE ... *New*	179	155
Source Print *by Aldebaran Labs*	97	75
TLIB *Version Control System by Burton*	100	89
Tree Diagrammer *by Aldebaran Labs*	77	67

raima products

dbQUERY *Single-User Query Utility*	195	129
Single-User with Source Code	495	389
Multi-User	495	389
Multi-User with Source Code	990	699
dbVISTA *Single-User DBMS*	195	129
Single-User with Source Code	495	389
Multi-User	495	389
Multi-User with Source Code	990	699

sco products

Complete XENIX System V *by SCO*	1295	994
Development System	595	499
Operating System *Specify XT or AT*	595	499
Text Processing Package	195	144
Lyrix *by SCO*	595	449
SCO Professional *1-2-3 Workalike for XENIX*	795	595
SCO XENIX-NET	595	495
XENIX System V 386 *by SCO* ... *New*	CALL	CALL

softcraft products

Btrieve *ISAM Mgr with No Royalties*	245	184
Xtrieve *Query Utility*	245	184
Report Option *for Xtrieve*	145	99
Btrieve/N *for Networks*	595	454
Xtrieve/N	595	454
Report Option/N *for Xtrieve/N*	345	269

text editors

Brief & dBrief Combo *from Solution Systems*	275	CALL
Brief	195	CALL
dBrief *Customizes Brief for dBASE III*	95	CALL
de *by David Livshin* ... *New*	75	65
Epsilon *Emacs-like editor by Lugaru*	195	147
KEDIT *by Mansfield Software*	125	98
Micro/SPF *by PHASER SYSTEMS*	175	139
Microsoft Word ... *New Version*	450	269
PC/VI *by Custom Software Systems*	149	99
SPF/PC *by Command Technology Corp*	CALL	CALL
Vedit Plus *by CompuView*	185	128

turbo pascal utilities

ALICE *Interpreter by Software Channels*	95	66
DOS/BIOS & Mouse Tools *by Quinn-Curtis*	75	67
Flash-up Windows *by Software Bottling*	90	78
MACH 2 *for Turbo Pascal by Micro Help*	69	55
MetraByte D/A Tools *by Quinn-Curtis*	100	89
Science & Engrg Tools *by Quinn-Curtis*	75	67
Screen Sculptor *by Software Bottling*	125	91
Speed Screen *by Software Bottling*	35	32
System Builder *by Royal American* ... *New Version*	150	129
IMPEX *Query Utility*	100	89
Report Builder ... *New Version*	130	115
TDebugPLUS *by TurboPower Software*	60	49
Tmark *by Tangent Designs*	80	69
Turbo EXTENDER *by TurboPower Software*	85	64
Turbo OPTIMIZER *by TurboPower*	75	65
with Source Code	125	98
Turbo Professional *by Sunny Hill*	70	45
TurboHALO *from IMSI*	129	98
TurboPower Utilities *by TurboPower*	95	78
TurboRef *by Gracon Services*	50	35
TURBOsmith *Source Debugger by Visual Age*	99	89
Universal Graphics Library *by Quinn-Curtis*	130	119

wendin products

Operating System Toolbox	99	79
PCNX *Operating system*	99	79
PCVMS *Similar to VAX/VMS*	99	79
Wendin-DOS *Multitasking DOS*	99	85
Wendin-DOS Application Developer's Kit ... *New*	99	85
XTC *Text Editor w/Pascal source*	99	75

xenix/unix products

Btrieve *ISAM File Mgr by SoftCraft*	595	454
C-terp *by Gimpel, Specify compiler*	498	379
c-tree *ISAM Mgr by FairCom*	395	315
dBx *with Library Source by Desktop AI*	550	419
DIRECTORY SHELL 286		
by American Mgmt Sys ... *New*	349	295
DIRECTORY SHELL 386		
by American Mgmt Sys ... *New*	495	415
DOSIX *Console Version by Data Basics*	400	349
DOSIX *User Version by Data Basics*	200	179
Micro Focus Products *See Micro Focus Section*		
Microport Products *See Microport Section*		
Microsoft Products *See Microsoft Section*		
PANEL Plus *by Roundhill Computer Systems*	795	535
REAL-TOOLS *Binary Version by PCT*	99	89
Library Source Version	599	539
Complete Source Version	999	729
RM/COBOL *by Ryan-McFarland*	1250	949
RM/FORTRAN *by Ryan-McFarland*	750	549
SCO Products *See SCO Section*		

Terms are subject to change.

©1987 Programmers Connection, Inc.

CIRCLE 29 ON READER SERVICE CARD

LOWEST PRICES

Due to printing lead times, some of our current prices may differ from those shown here. Call for latest pricing.

FREE SHIPPING

Orders within the USA (including Alaska & Hawaii) are shipped FREE via UPS. Express shipping is available at the shipping carrier's standard rate with no rush fees or handling charges. To avoid delays when ordering by mail, please call first to determine the exact cost of express shipping.

VISA and MasterCard are accepted at no extra cost. Your card is charged when your order is shipped. Mail orders please include credit card expiration date and authorized signature.

CODs AND POs

CODs and Purchase Orders are accepted at no extra cost. No personal checks are accepted on COD orders. POs with net 30-day terms (with initial minimum order of $100) are available to qualified US accounts only.

SALES TAX

Orders outside of Ohio are not charged state sales tax. Ohio customers please add 6% Ohio tax or provide proof of tax-exemption.

INTERNATIONAL ORDERS

Shipping charges for International and Canadian orders are based on the shipping carrier's standard rate. Since rates vary between carriers, please call or write for the exact cost. International orders (except Canada), please include an additional $10 for export preparation. All payments must be made with US funds drawn on a US bank. Please include your telephone number when ordering by mail. Due to government regulations, we cannot ship to all countries.

VOLUME ORDERS

Volume orders may qualify for additional discounts. Call us for special pricing.

SOUND ADVICE

Our knowledgeable technical staff can answer technical questions, assist in comparing products and send you detailed product information tailored to your needs.

30-DAY GUARANTEE

Most of our products (excluding books) come with a 30-day documentation evaluation period or a 30-day return guarantee. Please note that some manufacturers restrict us from offering guarantees on their products. Call for more information.

MAIL ORDERS

Please include your telephone number on all mail orders. Be sure to specify computer, operating system and any applicable compiler or hardware interface(s). Send mail orders to:

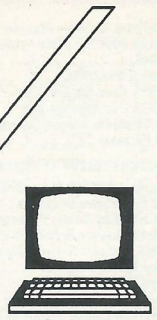
Programmer's Connection
7479 Whipple Ave. NW
North Canton, OH 44720

USA	**800-336-1166**
CANADA	**800-225-1166**
OHIO & ALASKA (Collect)	**216-494-3781**
TELEX	**9102406879**
EASYLINK	**62806530**
INTERNATIONAL	**216-494-3781**
CUSTOMER SERVICE	**216-494-8899**

Hours: Weekdays 8:30 AM to 8:00 PM EST.

PROGRAMMER'S CONNECTION — QUALITY INTEGRITY Service SUPPORT PRICE

PANEL® *Plus*



Advanced Screen Manager

Building an interactive application in C? PANEL *Plus* provides the features you need for professional program development:

PRODUCTIVITY

The PANEL *Plus* interactive screen design tools are the fastest way to get your application screens set out and tested. Just type prompts on the screen, mark out entry fields, define display and entry attributes, help boxes, borders, pop-up areas. Fields can be edited, moved and resized, and validation details entered – with the screen displayed so that you can see the effect of your changes.

Then PANEL *Plus* saves your work to disk, from where you can either load the design directly into your application program, or for better control, automatically generate C data structures, field areas, and header files which are compiled and linked into your program.

QUALITY

PANEL *Plus* screens can include all the features demanded by today's applications. Several different menu types are provided, including highlighted bars with help lines. Easy-to-use library functions support pop-up fields, horizontal and vertical scrolling in a field, and validation exits for supplied or custom data checking functions. Text functions can also be carried out in graphics mode using a supported graphics function library.

EASE OF USE

Although the library contains over 150 functions, it is logically organised so that most programs will only need to use a small subset. Users of Roundhill's PANEL package will find that the old calls have been emulated (and all screens are compatible). New, expanded documentation is provided with examples of all the main function calls. PANEL *Plus* includes full library source, with variant files for all supported systems, and no royalties are payable for the use of PANEL *Plus* libraries when linked into user applications.

PORTABILITY

PANEL *Plus* is designed to allow your programs to be ported to just about any environment where you can find a C compiler. Every version of PANEL *Plus* includes source modules for interfacing to all of the following environments: MS-DOS/PC-DOS (portable and memory-mapped), OS/2 protected mode, Amiga Intuition, Unix (with and without *termcap* or *termio*), Xenix, DOS-J (including Asian 16-bit character editing), and VAX/VMS. Graphics libraries supported include MetaWindow, HALO, Essential Graphics, and Microsoft C V5. The Microsoft mouse can be used in PC versions.

PANEL *Plus* for MS-DOS, with full library source, is priced at $495.00. Versions are available for the Manx Aztec, Borland, Lattice, MetaWare, Microsoft, and Wizard C compilers. Please call for prices of PANEL *Plus* for Xenix and Unix systems, and for VAX/VMS. Existing registered users of PANEL will receive a credit against the PANEL *Plus* license fee.

New !

Roundhill announces two screen tools for use with the exciting new C compilers from Borland and Microsoft:

PANEL/TC – $129.00
For use with Borland's Turbo C

PANEL/QC – $129.00
For use with Microsoft's Quick C

Each package is configured for use on an IBM PC or compatible system, and screens can be designed and built into your programs while running in the special development environment provided with the compiler.

All the PANEL *Plus* library functions are supported, and source code for every validation function is provided so that you can customise the entry checking to suit your application. When you need to move your programs to other environments, or need the full library source for other reasons, upgrades to PANEL *Plus* are available.

PANEL/QC can be run in any of the graphics modes supported by Quick C, and also interfaces to Microsoft C V5.

Roundhill Computer Systems Limited **(201) 569 2265**
PO Box 8107, Englewood NJ 07631

Roundhill Computer Systems Limited **(0672) 54675**
PO Box 14 Marlborough SN8 1LR England

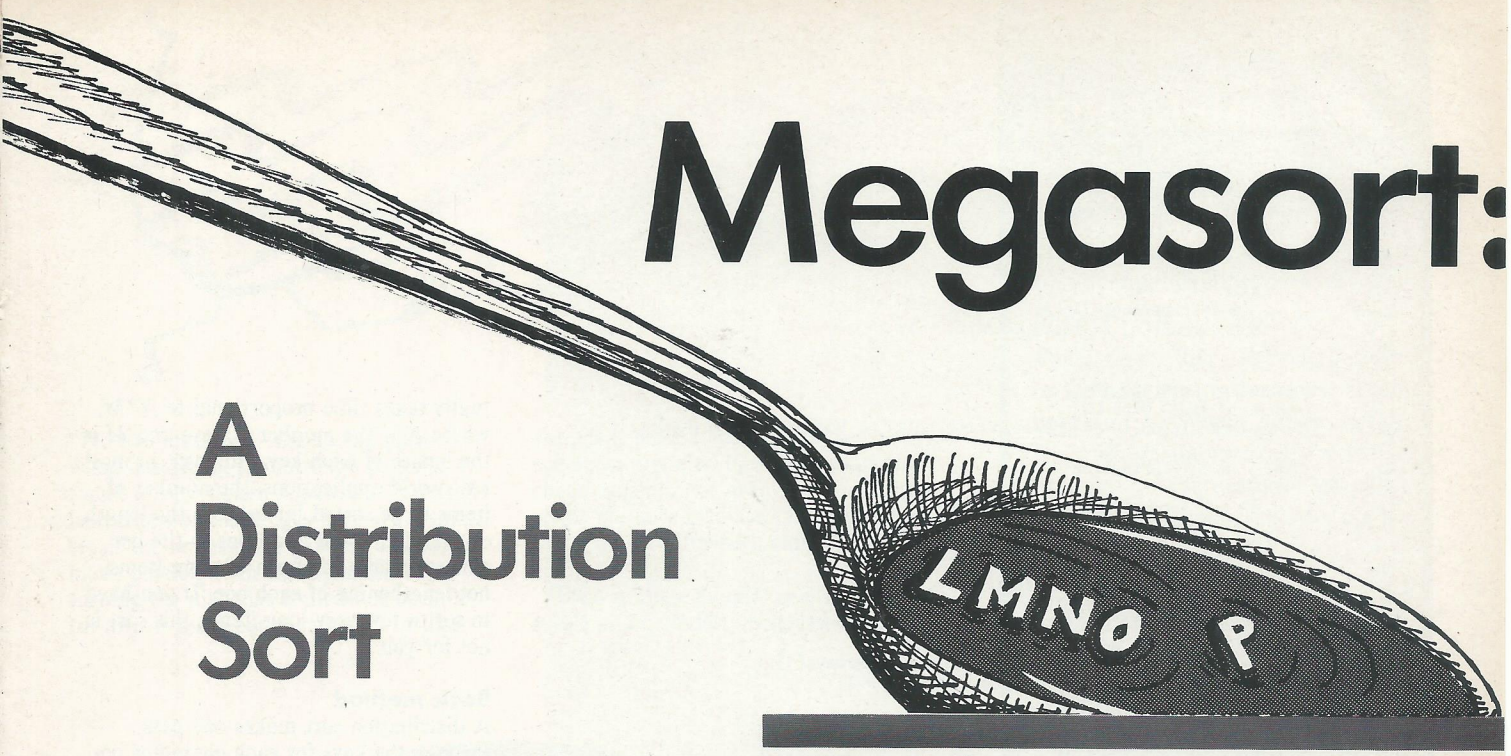
Telex (UK): 444453 AWARE G **BIX:** *join roundhill*
Fax (UK): (0672) 54436

Roundhill Computer Systems

CIRCLE 30 ON READER SERVICE CARD

Megasort:

A Distribution Sort



By Steve Heller

A few months ago, while writing a fairly large Turbo Pascal program, I needed to sort a couple of lists of a few hundred to a few thousand keys of 10 characters into alphabetical order on an IBM PC compatible. My first impulse was to use the old reliable bubble sort. After all, how long could it take to sort such relatively short lists? Bubble sorts are extremely inefficient, and I certainly would not use one for a large sorting task, but this task didn't seem large to me.

For 1,000 keys, the processing took approximately 15 minutes—on a 7 MHz processor. Since I wanted to run the program repeatedly on slightly differing sets of data, I needed a considerably faster sort.

Of course, many sorting techniques are faster than the bubble sort; in fact, just about every other sort is faster. However, most of them are complicated to program, and I wanted something I could implement quickly and easily in Turbo Pascal.

Luckily, a sort does exist that is very simple to implement and still runs extremely quickly. In fact, its execution time is linear in the number of keys to be sorted. It is no more than a constant factor slower than any other sort, faster than almost any other on large arrays, and the simplest sort of all, in my opinion.

The sort I'm referring to is called a distribution sort and is described in Donald Knuth's *The Art of Computer Programming: Sorting and Searching*, Vol. 3, under the heading "distribution counting." It is one of three sorting techniques Knuth lists that are linear in the number of items to be sorted. The other two are radix exchange and radix list sort, both of which are more complicated to program, especially in a higher-level language.

As you can see in Listing 1, the actual code of the sort is only about 60 lines of Turbo Pascal. The time to do the sort depends both on the number of keys and the length of each key. The length of the keys is important because this program makes one pass over the input data for each character position of the keys to be sorted.

On each pass, the list is ordered by one more character position than on the previous pass. Therefore, this method ac-

```
 1: PROGRAM megaa; {Copyright 1986 by Steve Heller, Inc.  All rights reserved.}
 2:
 3: CONST
 4:   MaxSize = 5000;
 5:
 6: TYPE
 7:   AnyString = String[255];
 8:   SomeString = String[10];
 9:   StrPtrArr = ARRAY [1..MaxSize] OF ^AnyString;
10:   SortArray = ARRAY [Char] OF Integer;
11:
12: VAR
13:   TestArray : ^StrPtrArr;
14:   TestArray2 : ^StrPtrArr;
15:   TestArray3 : ^StrPtrArr;
16:   junk : AnyString;
17:   i : Integer;
18:   infile : text[10000];
19:   infilename : AnyString;
20:   outfile : text[10000];
21:   outfilename : AnyString;
22:   KeyLen : Integer;
23:   ArrayLength : Integer;
24:
25: PROCEDURE Megasort(VAR PtrArray:StrPtrArr; VAR SubArray1:StrPtrArr;
26:                    VAR Subarray2:StrPtrArr;
27:                    ArrayCount:Integer;KeyLength:Integer;ArraySize:Integer);
28:
29: VAR
30:   l : Char;
31:   m : Char;
32:   i : Integer;
33:   j : Integer;
34:   BucketCount  : SortArray;
35:   BucketPosition : SortArray;
36:   TempPtrArr : ^StrPtrArr;
37:   TempSubArr1: ^StrPtrArr;
38:   TempSubArr2: ^StrPtrArr;
39:
40:
```

Listing 1. *(Continued on page 64.)*

```
41: BEGIN
42:
43:   New(TempPtrArr);
44:   New(TempSubArr1);
45:   New(TempSubArr2);
46:
47:   FOR i := KeyLength DOWNTO 1 DO
48:     BEGIN
49:       FOR l := #0 TO #255 DO
50:         BucketCount[l] := 0;
51:       FOR j := 1 TO ArraySize DO
52:         BEGIN
53:           IF i > length(PtrArray[j]^) THEN
54:             m := #0
55:           ELSE
56:             m := PtrArray[j]^[i];
57:           BucketCount[m] := BucketCount[m] + 1;
58:         END;
59:
60:       BucketPosition[#0] := 1;
61:       FOR l := #1 TO #255 DO
62:         BucketPosition[l] := BucketCount[pred(l)] + BucketPosition[pred(l)];
63:
64:       FOR j := 1 TO ArraySize DO
65:         BEGIN
66:           IF i > length(PtrArray[j]^) THEN
67:             m := #0
68:           ELSE
69:             m := PtrArray[j]^[i];
70:           TempPtrArr^[BucketPosition[m]] := PtrArray[j];
71:           IF ArrayCount >=2 THEN
72:             TempSubArr1^[BucketPosition[m]] := SubArray1[j];
73:           IF ArrayCount =3 THEN
74:             TempSubArr2^[BucketPosition[m]] := SubArray2[j];
75:           BucketPosition[m] := BucketPosition[m] + 1;
76:         END;
77:
78:       FOR j := 1 TO ArraySize DO
79:         BEGIN
80:           PtrArray[j] := TempPtrArr^[j];
81:           IF ArrayCount >=2 THEN
82:             SubArray1[j] := TempSubArr1^[j];
83:           IF ArrayCount = 3 THEN
84:             SubArray2[j] := TempSubArr2^[j];
85:         END;
86:
87:     END;
88:
89:     Dispose(TempPtrArr);
90:     Dispose(TempSubArr1);
91:     Dispose(TempSubArr2);
92:
93:   END;
 98: BEGIN
 99:   New(TestArray);
100:
101:   Write('Input file name: ');
102:   ReadLn(infilename);
103:   Write('Output file name: ');
104:   ReadLn(outfilename);
105:   Write('Key length: ');
106:   ReadLn(KeyLen);
107:   Assign(infile,infilename);
108:   Reset(infile);
109:   Assign(outfile,outfilename);
110:   Rewrite(outfile);
111:
112:   WriteLn('Reading input file.');
113:
114:   i := 0;
115:   WHILE NOT EOF(infile) DO
116:     BEGIN
117:       i := i + 1;
118:       ReadLn(infile,junk);
119:       GetMem(TestArray^[i],length(junk)+1);
120:       TestArray^[i]^ := junk;
121:     END;
122:
123:   ArrayLength := i;
124:
125:   WriteLn('Sorting.');
126:
127:   Megasort(TestArray^,TestArray^,TestArray^,1,KeyLen,ArrayLength);
128:
129:   WriteLn('Writing output file.');
130:
131:   FOR i := 1 TO ArrayLength DO
132:     WriteLn(outfile,TestArray^[i]^);
133:
134:   Close(infile);
135:   Close(outfile);
136:
137:   WriteLn('Done.');
138:
139: END.
```

Listing 1. *(Continued from page 63.)*

tually takes time proportional to $N*M$, where N is the number of keys and M is the length of each key. However, in most real-world applications, the number of items to be sorted far exceeds the length of each item, and additions to the list generally take the form of more items, not lengthening of each one. If you have to sort a few very long items, this sort is not for you.

Basic method

A distribution sort makes one pass through the keys for each character position in the key to discover how many keys have each possible ASCII character in the character position we are currently considering. The sort then makes another pass to rearrange the keys.

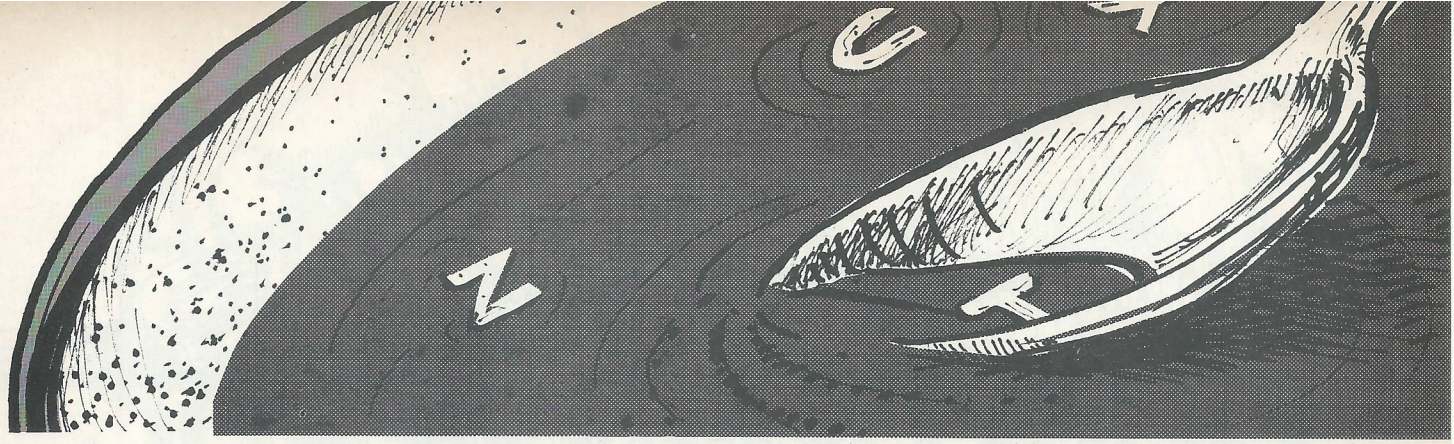
As a simplified example, suppose you have 10 keys to be sorted: three keys that begin with the letter A, five that begin with B, and two that begin with C—and you want to sort only on the first letter of each key. In this case, you can determine that the first key that starts with an A should be the first key in the result array, the second A key should be the second key, and the third A key should be the third key since all of the A keys should precede the B keys. The first key that starts with a B should be the fourth key, the second through fifth B keys should be the fifth through eighth keys, and the C keys should be numbers nine and 10 since all of the B keys should precede the C keys.

After describing the parameters and local variables needed for the *Megasort* procedure, I will go into more detail about the usual case in which the keys contain more than one character to be used in the sort.

Variables

As you can see from Listing 1, the *Megasort* procedure has six parameters. The first one, *PtrArray*, is the array of pointers to the sort keys. This array will be rearranged into ASCII character code order.

The second and third arguments, *SubArray1* and *SubArray2*, are auxiliary arrays that will be rearranged in the same way as the sort key array. This enables



you to keep several arrays in order by using one key.

For example, you might want to sort personnel records by last names of the employees. You would extract the last names and use them to form the sort key array. The auxiliary array would be pointers to the complete records. You would have to modify the procedure declaration to be able to use record arrays rather than string arrays for the auxiliary arrays. Of course, these first three parameters must be passed by address (*VAR* parameters) rather than by value so they can be modified during the sort.

The fourth parameter, *ArrayCount*, controls how many of the three possible arrays mentioned are actually processed during the sort. A value of 1 means only the key array will be processed, a value of 2 means the key array and one auxiliary array will be processed, and a value of 3 means the key array and both auxiliary arrays will be processed.

This parameter is necessary because Pascal does not allow procedures to take a variable number of arguments. If either or both of the auxiliary arrays will not be used, their argument positions in the call must be filled in with dummy array names of the same type, and the fourth parameter must be set to the appropriate value.

The fifth parameter, *KeyLength*, is the number of characters of each key that will be considered in the sort, starting from the left end of the key. This parameter enables you to limit the number of character positions to the number that is actually significant for your purposes. Any characters beyond this limit will be ignored in the sort, and any keys shorter than this limit will be padded on the right with nulls, ensuring that they will precede any longer keys that match them up to their ends in the final output.

The sixth parameter, *ArraySize*, is the number of keys to be sorted. *ArraySize* is necessary because Pascal requires the size of an actual array parameter to match the size of the formal array parameter. Therefore, you must use this parameter to limit the number of keys to be sorted to the number of valid entries in the key array.

In addition to the arguments, a number of local variables are required. The most important of these are *BucketCount* and *BucketPosition*, which are used in the calculation of the number of keys that fall in each distribution category, or bucket, and the three *Temp* arrays, which are used to hold the key pointers temporarily while they are being rearranged.

Megasort code

The code for the *Megasort* procedure begins at line 43 in Listing 1, calling the *New* procedure to allocate temporary arrays for use in the sorting process. The *i* loop that begins on line 47 counts backward from the last character of the key to the first, so that the sort proceeds from right to left. The right-to-left procedure ensures that two keys having the same key character in the current pass will maintain their order after the pass.

For example, if you are sorting the two keys ABC and ABD and are currently processing the second character position, the positions of ABC and ABD will not be exchanged. Therefore, if you sort from the right end first, after the first pass all the keys will be in order by last characters. After the second pass they will be in order by next-to-last and last characters, and so on.

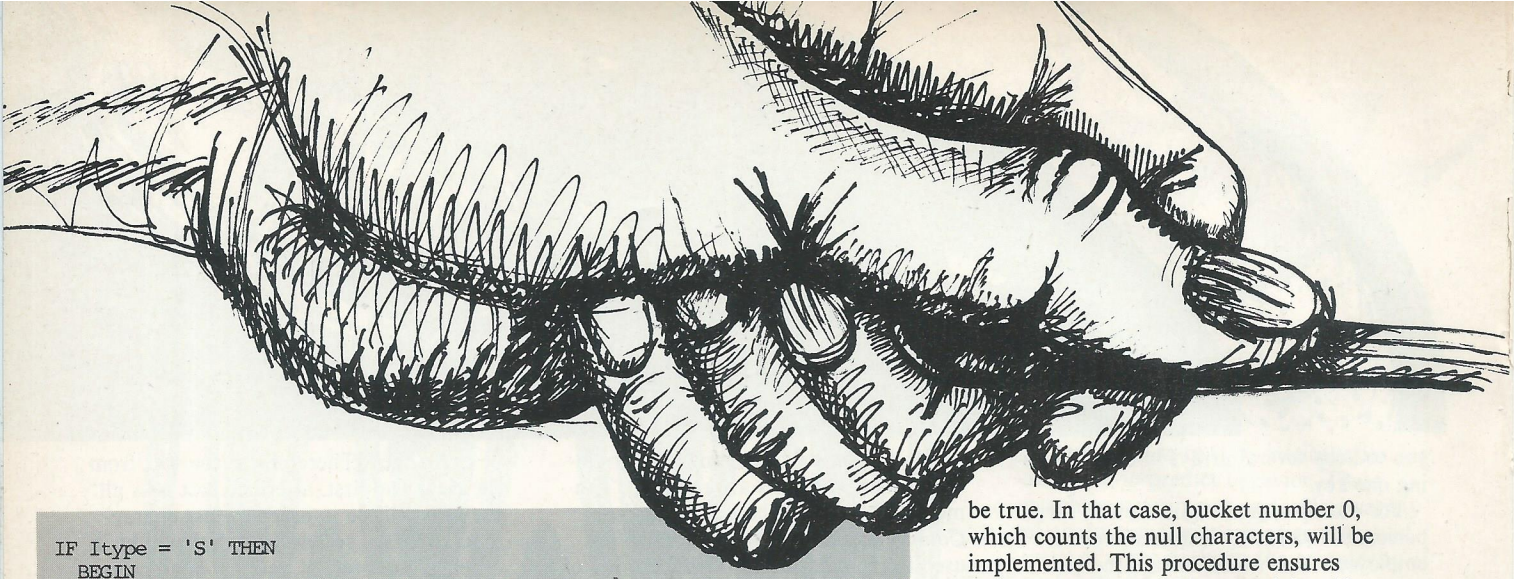
The code to handle each character position in the keys starts at line 49. Lines 49-50 clear the sort array *BucketCount*, which keeps track of how many keys have each ASCII character code in the current position.

For example, *BucketCount['1']* keeps track of the number of keys that have the character 1, and *BucketCount['A']* does the same for the keys that have an A in the current sort position.

Lines 51-58 accumulate the key counts mentioned in the preceding paragraph. The *j* loop starting on line 51 steps through all the keys. The *IF* statement on lines 53-56 determines the bucket number that will be incremented to account for the key currently being processed. If the character position currently being processed is past the end of the current key, the expression on line 53 will

```
{SORTDAT.PAS - generates sort data for MEGASORT testing}
{861223 :2200}

VAR
  i,j : Integer;
  ir  : Real;
  s : String[255];
  t : Text[10000];
  n : Real;
  Itype : Char;
  MaxLength : Integer;
  Ran : Char;
  RealTemp : Real;
  IntTemp : Integer;
  RealExp : ARRAY [-30..30] OF Real;
  FName   : String[80];
BEGIN
  RealExp[-30] := 1E-30;
  FOR i := -29 TO 30 DO
    RealExp[i] := RealExp[i-1]*10;
  Write('Name of data file to be generated: ');
  ReadLn(FName);
  Write('Number of items to generate: ');
  ReadLn(n);
  Write('Type of items (R for real, I for integer, S for string): ');
  ReadLn(Itype);
  Itype := Upcase(Itype);
```

Listing 2. *(Continued on page 66.)*



```
IF Itype = 'S' THEN
  BEGIN
    Write('Maximum length of strings: ');
    ReadLn(MaxLength);
    Write('Random string length or all maximum length (R or M): ');
    ReadLn(Ran);
    Ran := Upcase(Ran);
  END;

Assign(t,Fname);
Rewrite(t);
ir := 1.Ø;
REPEAT
  BEGIN
    IF ir = 1ØØØ*int(ir/1ØØØ) THEN WriteLn(ir:1Ø:Ø);
    IF Itype = 'S' THEN
      BEGIN
        s := '';
        IF Ran = 'R' THEN
          FOR j := 1 TO random(MaxLength) DO
            s := s + chr(random(64)+32)
        ELSE
          FOR j := 1 TO MaxLength DO
            s := s + chr(random(64)+32);
        WriteLn(t,s);
          END
        ELSE IF Itype = 'R' THEN
          BEGIN
            RealTemp := Random;
            IF Random > Ø.5 THEN
              RealTemp := -RealTemp;
            IntTemp := Random(3Ø);
            RealTemp := RealTemp * RealExp[IntTemp];
            Str(RealTemp,s);
            IF RealTemp > Ø THEN
              s := copy(s,3,length(s))
            ELSE
              s := copy(s,2,length(s));
            WriteLn(t,s);
          END
        ELSE IF Itype = 'I' THEN
          BEGIN
            IntTemp := Random(32767);
            IF Random >Ø.5 THEN
              IntTemp := -IntTemp;
            Str(IntTemp,s);
            WriteLn(t,s);
          END;
      END;
      ir := ir + 1.Ø;
    UNTIL ir > n;
    Close(t);
  END.
```

Listing 2. *(Continued from page 65.)*

be true. In that case, bucket number 0, which counts the null characters, will be implemented. This procedure ensures that shorter keys will be sorted first if their first characters are the same as those of a longer key.

For instance, XYZABC will be sorted before XYZABCA if the *KeyLength* parameter is at least 7, because XYZABC is effectively padded on the right with at least one null character.

On the other hand, if the character position currently being processed is within the length of the key currently being processed, the assignment on line 56 will select the bucket to be incremented according to the ASCII value of the *i*th character of the *j*th key.

If you are processing the second character position and the current key is "The chip is fast," *BucketCount['h']* will be selected to be incremented. Line 57 increments the bucket selected by the *IF* statement.

After the *j* loop from lines 51-58 has been executed for each key in the key array, the buckets in *BucketCount* contain the counts for each possible character value in the keys (for the current character position).

However, you need the cumulative count of keys for each character value. That is, suppose you want to sort the following one-character keys:

C
B
A
D
B
D
C
A
B
D

Of the 10 keys, you have the following bucket distribution:

Bucket	A	B	C	D
Counts	2	3	2	3

But you really want to know where the As start, where the Bs start, etc. The table you want looks like this:

TE - TEXT EDITOR

If you own an IBM PC or a compatible personal computer, and if you are searching for a powerful, feature-packed text editor, your search is over. We have now made available just that text editor for a price unheard of before. Now for $45.00 you can own a text editor which gives the best performance for your dollars.

The features include:

* Complete full screen editor.
* Support for multiple files.
* UNDO of delete commands.
* Fast file loading and unloading.
* Replace with verify.
* Print from within the text editor.
* Block oriented change and print commands.
* Execute DOS commands or temporarily exit to DOS.
* Import a file at current location.
* Export a block of text to a file.
* Import a block of text from a secondary file.
* Uses entire memory for text editing.
* Many other miscellaneous features.

TE Developer's Kit: *TE developers kit includes the source code for the text editor, all the required Library routines to link the editor and many other miscellaneous routines. Written in Microsoft 'C', it is the most cost effective way of incorporating text editing feature into your applications.*
The kit is most favorably priced at $95.00

To place an order, call
Programmers Connection
1-800-336-1166

Sub Systems, Inc.
159 Main Street, #8C
Stoneham, MA 02180

CIRCLE 39 ON READER SERVICE CARD

SOFTWARE PROFESSIONALS

In Less Than Two Decades, We've Become a World Leader In Problem-Solving Software Systems.

Since 1970, IMSL's international reputation for excellence in the development and distribution of scientific, engineering and statistical software has made us a world leader. Come join us at our Houston headquarters.

These positions involve developing software application packages used in mathematics, engineering and statistics. Candidates should have a Master's degree in computer science, strong FORTRAN programming ability, excellent communication skills, and a software background in:

- Design of programming languages
- Compiler design and construction
- Development and maintenance of large-scale FORTRAN application systems.

IMSL is a growing company providing challenging opportunities, competitive salaries and excellent benefits. Send resume and salary requirements, to **Personnel, Dept. SDP004-CL, IMSL, Inc., 2500 ParkWest Tower One, 2500 CityWest Boulevard, Houston, Texas 77042-3020.**
Equal Opportunity Employer.

IMSL

CIRCLE 40 ON READER SERVICE CARD

Parallel Programming for "C"

INTERWORK
A Concurrent Programming Toolkit

Interwork is a "C" program library which allows you to write your programs as a set of cooperating concurrent tasks. Very useful for simulation, real-time applications, and experimentation with parallel programming.

FEATURES

- Supports a very large number of tasks (typically more than 100) limited only by available memory. Low overhead per task results in very fast context switching.
- Provides a full set of inter-task communication (ITC) facilities, including shared memory, locks, semaphores, blocking queues, and UNIX*-style signals. Also has building blocks for constructing your own ITC facilities.
- Handles interrupts (DOS version) and integrates them into task scheduling. Supply your own interrupt handlers or block tasks on interrupts.
- Lets you trace task switches and inter-task communication.
- Comes with complete documentation including a user's manual and reference manual of commands.

Interwork is available for the following systems:

Hardware	*Operating System*	*Price*
IBM PC, XT, AT	PC-DOS 2.0 or later	$129
IBM PC AT	XENIX*	$159
DEC VAX*, SUN	UNIX 4.2BSD	$249

PC-DOS version is compatible with DeSmet, Lattice, and Microsoft C compilers.
Please specify hardware and operating system when ordering. Shipping and handling included; COD orders add $2.50. Send check or money order to:

Block Island Technologies
Innovative Computer Software
13563 NW Cornell Road, Suite 230, Portland, Oregon 97229-5892
(503) 241-8971

*Trademarks: UNIX, AT&T Bell Laboratories, Inc.; XENIX, Microsoft, Inc.; VAX, Digital Equipment Corporation

CIRCLE 41 ON READER SERVICE CARD

Announcing WKS LIBRARY

NEW!

The Lotus "Wrap-Around" for C Programs

Now you can write and read Lotus worksheets directly from a C program!

WKS LIBRARY lets you:

- avoid time-consuming file translation steps
- control the execution of 1-2-3 from inside your application
- use Lotus as a data entry screen in your C program
- generate "live" financial statements with formulas for totals

Feature this:

- reads and writes .WKS, .WK1 and .WR1 worksheets
- writes integers, floats, strings, formulas, macros and dates using *wprintf()*
- reads using *wscanf()*, converting column contents to C variables according to format specs
- has low-level functions for manipulating individual cells
- provides Lotus control functions such as: formats, column widths, initial cursor position, range names, cell protection
- supports Lattice, Microsoft & Turbo compilers for DOS, plus most UNIX environments

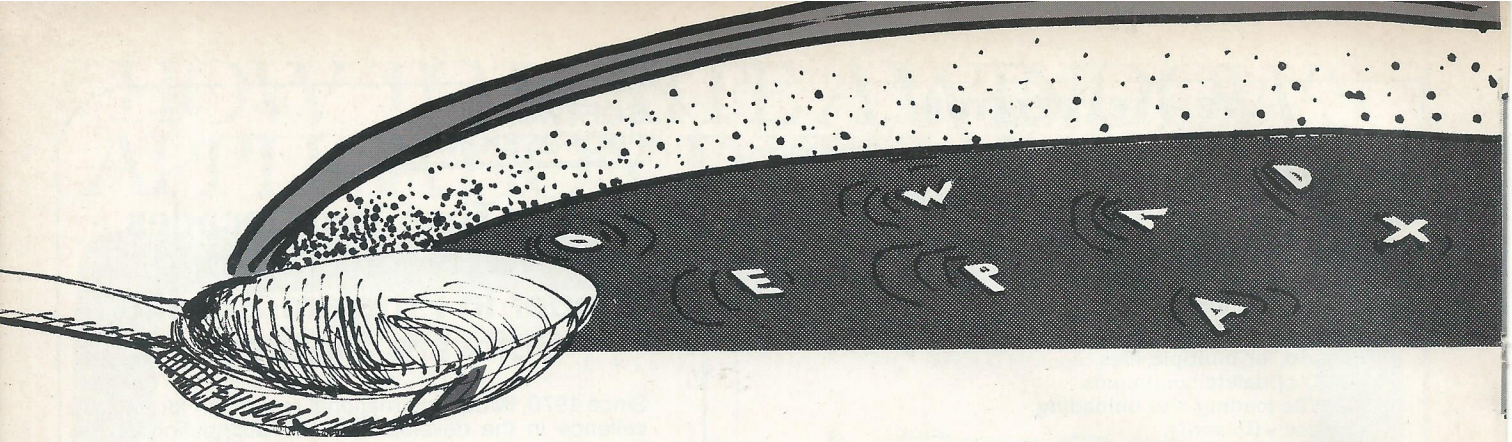
Source Code provided **No Royalties on executable programs**

Only $89
(includes free 800-line support)

VISA MasterCard

ORDER TODAY!
Toll-free
(800) 367-9882

Tenon Software, Inc.
1980 - 112th NE, #250, Bellevue, WA 98004
(206) 453-1914 (in Washington state)



Bucket	A	B	C	D
Positions	1	3	6	8

Now you know that you should move the first A to the first position in the output, the first B to the third position, the first C to the sixth position, and the first D to the eighth position. How is this table derived?

The Bucket positions table is computed by the following procedure:

- (Line 60) The first output always goes to output position 1.
- (Line 61) For all the rest of the bucket positions, *do*.
- (Line 62) Each bucket position starts at the previous bucket position plus the number of items in the previous bucket.

Now you are ready to move the key pointers (and the auxiliary array pointers) to their respective places in the output of this sorting pass. The *j* loop starting on line 64 does this. First, on lines 66-69, it computes which bucket of the *BucketPosition* array should be used for the output addressing.

As in the initial determination of the *BucketCount* array (lines 51-58), if a key value's length is less than the character position you are currently processing, the key is assigned to bucket #0. Otherwise, the *BucketPosition* element is determined by the ASCII value of the character at the current position in the key.

Then, in line 70, the key pointer is copied to the position in *TempPtrArr* that corresponds to its *BucketPosition* entry. Lines 71-74 do the same for the auxiliary arrays that need to be sorted along with the key array. Line 75 increments the *BucketPosition* entry so the next key that uses that entry will use the next position in the output array rather than writing over the position you just used.

Then, lines 78-85 copy the output pointer arrays back to the main arrays, in preparation for the next sorting pass (or completion of the sort, if this is the last pass). Finally, lines 89-91 release the temporary storage arrays used by *Megasort*.

The remainder of the code in Listing 1 is the main program, which can be used to sort files that contain one record per line of ASCII data if the sort key starts at the beginning of the line. You can use the SORTDAT.PAS program (Listing 2) to create a file of random ASCII text to test the sort. If you are using the program from Listing 1, you will find that the time taken by the sort itself (from the "Sorting." message to "Writing output file.") will be only 42 seconds for 5,000 10-byte keys on a 4.77 MHz 8088 machine such as an IBM PC.

I have written an InLine-accelerated version of Megasort, which can do the same sort in only six seconds, as well as a C version and a 68000 assembly language version. Anyone interested in obtaining these versions of the sort, or in using the version presented here for commercial purposes, should contact me at Chrysalis Software Corp., P.O. Box 0335, Baldwin, N.Y. 11510, (516) 868-2698.

I am not claiming that Megasort is the fastest sort possible on any kind of data. If the keys are long (greater than 10 bytes) and their distribution is known, then other sorts are competitive, or in some cases faster (for example, address calculation sorting). However, Megasort is very simple to code in any language and is at least as fast as any other sort for short keys.

Another important factor in comparing the speeds of the three sorting methods analyzed in Table 1 is that Unshuffle and Quicksort are data-dependent (as are most other sorts). Their worst cases are much worse than the average cases given in Table 1. According to Knuth, the average case for Quicksort is:

$$10.63 * N * \ln(N) + 2.11 * N$$

or 10 seconds for $N = 100000$, assuming an instruction time of 1 microsecond. The worst case is given as:

$$>= N*N$$

or at least 2.78 hours, for the same N. Megasort does not have this drawback, as its timing is data-independent; the figures given are valid for all possible data sets.

The times given in Table 1 were provided by Bob Gebelein, who has coded all of the sorts in 68000 assembly language and calculated their timing from 68000 cycle counts. The key lengths specified at the top apply only to Megasort, as timings of the other sorts are relatively independent of key length. ■

Steve Heller is president of Chrysalis Software Corp., Baldwin, N.Y. He has been programming for the last 18 years, and graduated from Shimer College, Waukegan, Ill., in 1970.

Artwork: Mark Duran

Comparing the three sorts (sec)

Number of records	Unshuffle	Quicksort	Megasort k=2	Megasort k=3	Megasort k=4	Megasort k=5	Megasort k=10	Megasort k=15
500	.33	.15	.04	.05	.07	.10	.19	.28
1,000	.91	.34	.08	.10	.14	.18	.35	.52
5,000	10.17	2.16	.38	.53	.69	.86	1.64	2.43
10,000	29.06	4.68	.76	1.06	1.37	1.70	3.26	4.82
15,000	56.05	7.16	1.14	1.59	2.06	2.54	4.88	7.22
50,000	428.72	27.21	3.77	5.31	6.87	8.43	16.19	23.96

Key lengths 2, 3, and 4 were illustrated because they reflect some very common sort keys. Key length 2 is, of course, a 16-bit integer, which is commonly used as a record number. 3 represents a telephone number or a ZIP code, coded as a binary number, and 4 represents a Social Security number, coded as a binary number.

Table 1.

What experts are saying about PC Scheme from Texas Instruments:

"...At less than $100, this is easily the best buy in Lisp systems for PCs."

Dr. Dobb's Journal, February 1987

Discover how powerful—and inexpensive—PC symbolic programming can be with PC Scheme from Texas Instruments. Whether you're an experienced Lisp programmer or just beginning, PC Scheme is the complete, $95* solution to your software development needs.

PC Scheme combines elegant simplicity with remarkable speed in a full Lisp development system. Named *PC Tech Journal*'s Product of the Month (August 1986), PC Scheme brings professional Lisp programming features to personal computers.

PC Scheme 3.0

- —Optimizing incremental byte-code compiler for ease of programming and operation
- —EMACS-like editor
- —Lexical scoping of variables
- —Ability to suspend PC Scheme, execute DOS-based programs, then return to PC Scheme
- —Random-file access and binary-file support
- —Extensions for debugging, graphics and windowing
- —External language interface to C, Turbo Pascal® and other languages
- —SCOOPS (Scheme Object-Oriented Programming System)
- —Two-megabyte extended/expanded memory support
- —New manuals with tutorials and examples

Find out for yourself why experts are praising PC Scheme. For the dealer nearest you, or to order by phone, call toll-free:

1-800-527-3500

** TI Suggested list price*

PC Scheme runs on IBM® Personal Computers and compatibles (including the Texas Instruments Business-Pro™ computer). Minimum configuration: 512K RAM, dual floppy system.

Turbo Pascal is a registered trademark of Borland International. IBM is a registered trademark of International Business Machines Corporation. Business-Pro is a trademark of Texas Instruments Incorporated.

TEXAS INSTRUMENTS

261933A

© 1987 Texas Instruments Incorporated

Programmer's Paradise Gives You Superb Selection, Personal Service and Unbeatable Prices!

Welcome to Paradise. The microcomputer software source that caters to your programming needs. Discover the Many Advantages of Paradise ...

- Lowest price guaranteed
- Latest versions
- Huge inventory, immediate shipment
- Knowledgeable sales staff
- Special orders
- 30-day money-back guarantee*

Corporate Buyers—Call for special discounts and benefits!

We'll Match Any Nationally Advertised Price

Call or write for the latest catalog

	LIST	OURS
386 SOFTWARE		
ADVANTAGE 386 C	895	**805**
ADVANTAGE 386 PASCAL	895	**805**
PC-MOS 386	CALL	**CALL**
PHARLAP 386|ASM/LINK	495	**419**
VM/386	195	**CALL**
386 MICROPORT SYSTEM/386	799	**699**
DOS/MERGE 386	345	**295**
386 SCO XENIX SYS V COMPLETE	1495	**1195**
APL		
APL* PLUS/PC	595	**429**
POCKET APL	95	**79**
SCREENACE	195	**169**
ARTIFICIAL INTELLIGENCE		
ACTIVE PROLOG TUTOR	65	**55**
ARITY STANDARD PROLOG	95	**79**
CHALCEDONY PROLOG	100	**89**
GOLDEN COMMON LISP	495	**455**
GCL 286 DEVELOPER	1190	**1095**
MICROSOFT LISP	250	**155**
PC SCHEME	95	**85**
PROLOG 86 PLUS	250	**CALL**
LPA PROLOG PRO COMPILER	895	**759**
SMALLTALK V	99	**85**
TRANS LISP PLUS	195	**165**
TURBO PROLOG	100	**65**
TURBO PROLOG TOOLBOX	100	**65**
VP EXPERT	100	**89**
ASSEMBLERS/LINKERS		
ADVANTAGE DISASSEMBLER	295	**265**
ADVANTAGE LINK	395	**359**
ASMLIB	149	**125**
DR ASSEMBLER + TOOLS	200	**179**
EZ___ASM	70	**65**
MICROSOFT MACRO ASSEMBLER	150	**95**
PASM86	195	**109**
PLINK86PLUS	495	**275**
RELMS CROSS ASSEMBLERS	CALL	**CALL**
UNIWARE CROSS ASSEMBLERS	CALL	**CALL**
VISIBLE COMPUTER 80286	100	**89**
QUELO 68000 X-ASM	595	**509**
BASIC		
DB/LIB	99	**89**
FINALLY!	99	**89**
FLASH-UP	89	**79**
INSIDE TRACK	65	**55**
MACH 2	75	**59**
MS QUICKBASIC	99	**65**
QBASE	89	**79**
QUICK-TOOLS	130	**109**
QUICKPAK	69	**59**
SCREEN SCULPTOR	125	**95**
STAY-RES	95	**75**
STRUBAS	495	**445**
TRUE BASIC	100	**69**
TURBO BASIC	100	**65**
DATABASE TOOLBOX	100	**65**
EDITOR TOOLBOX	100	**65**
TELECOM TOOLBOX	100	**65**
C++		
ADVANTAGE C++	495	**479**
PFORCE ++	395	**209**
C COMPILERS		
AZTEC C-COMMERCIAL	499	**CALL**
AZTEC C-DEVELOPERS	299	**CALL**
C86PLUS	497	**375**
DATALIGHT OPTIMUM-C	139	**105**
HIGH C	595	**CALL**
LATTICE C *SPECIAL PRICE*	500	**CALL**
LET'S C W/CSD DEBUGGER	125	**99**
MICROSOFT C	450	**269**
QUICK C	99	**65**
TURBO C	100	**65**
C INTERPRETERS		
C-TERP	300	**229**
INSTANT C	500	**379**
RUN/C	120	**79**
RUN/C PROFESSIONAL	250	**155**

	LIST	OURS
C LIBRARIES		
ASYNCH MANAGER	175	**135**
BASIC___C	175	**129**
C-FOOD SMORGASBORD	150	**95**
W/SOURCE CODE	300	**179**
C TOOLS PLUS	175	**135**
C UTILITY LIBRARY	185	**119**
C-XPERT	295	**249**
ESSENTIAL COMMUNICATIONS	185	**125**
COMMUNICATIONS PLUS	250	**189**
GREENLEAF FUNCTIONS	185	**125**
GREENLEAF COMM LIBRARY	185	**125**
GREENLEAF SAMPLER (TURBO C)	95	**79**
MULTI-C	149	**135**
PFORCE	295	**209**
RESIDENT C W/SOURCE	198	
THE HAMMER	195	**129**
TIMESLICER	295	**265**
W/SOURCE CODE	1000	**895**
TURBO C TOOLS	129	**99**
COBOL		
COBOLSPII	395	**329**
FPLIB	149	**129**
MICRO FOCUS COBOL PRODUCTS	CALL	**CALL**
MICROSOFT COBOL	700	**439**
MICROSOFT SORT	195	**129**
OPT-TECH SORT	149	**105**
REALCICS	995	**785**
REALIA COBOL	995	**785**
W/REALMENU	1145	**899**
RM/COBOL	950	**759**
RM/COBOL-85	1250	**999**
RM/SCREENS	395	**315**
SCREENIO	400	**CALL**
SCREENPLAY	175	**129**

MACINTOSH PRODUCTS

APL*PLUS/MAC	395	299
BASIC COMPILER	195	**139**
BASIC INTERPRETER	99	**69**
CHALCEDONY PROLOG	100	**89**
CONSULAIR MAC C (MAC II)	600	**509**
FORTRAN COMPILER	295	**209**
LIGHTSPEED C	175	**125**
LIGHTSPEED PASCAL	125	**89**
MAC C JR.	80	**59**
MAC PROLOG	395	**335**
MACTRAN 77	199	**179**
MASTERFORTH	350	**315**
MACNOSY V2	90	**69**
TURBO PASCAL	100	**65**
Z BASIC	90	**69**

	LIST	OURS
DBASE TOOLS		
CLIPPER	695	**459**
DBASE III PLUS	695	**429**
DBASE TOOLS FOR C	80	**65**
DBASE GRAPHICS FOR C	80	**65**
DBRIEF	95	**85**
DBC III	250	**169**
DBUG III	195	**179**
DBXL	139	**99**
DBPOWERPAK III	100	**89**
DB/RA	200	**179**
DFLOW	149	**125**
DB FIND	99	**89**
DOCUMENTOR	295	**249**
FOXBASE PLUS	395	**339**
GENIFER	395	**299**
QUICKCODE III PLUS	295	**239**
QUICKSILVER	599	**499**
TOM RETTIG'S LIBRARY	100	**85**
R&R	109	**49**
UI PROGRAMMER	295	**249**
DEBUGGERS		
386 DEBUG	195	**155**
ADVANCED TRACE-86	175	**119**
BREAKOUT	125	**89**
C-SPRITE	175	**119**
PERISCOPE I	345	**289**
PERISCOPE II	175	**139**

	LIST	OURS
PERISCOPE II-X	145	**105**
PERISCOPE III 8 MHZ	995	**825**
PERISCOPE III 10 MHZ	1095	**899**
TDEBUG PLUS	60	**49**
PFIX 86 PLUS	395	**209**
SOFTPROBE II/TX	750	**715**
XVIEW86	60	**49**
DISK/DOS/KEYBOARD UTILITIES		
BACK-IT	100	**89**
BOOKMARK	100	**79**
COMMAND PLUS	80	**69**
FASTBACK	175	**129**
INTELLIGENT BACKUP	150	**135**
KEYWORKS ADVANCED	299	**239**
NORTON COMMANDER	75	**55**
NORTON UTILITIES	100	**59**
ADVANCED NORTON UTILITIES	150	**89**
PDISK	145	**99**
Q-DOS II	70	**59**
QUAID ANALYZER	99	**89**
TASKVIEW	80	**59**
FILE MANAGEMENT		
ASMTREE	395	**339**
BTRIEVE	245	**185**
XTRIEVE	245	**185**
REPORT OPTION	145	**99**
BTRIEVE/N	595	**455**
XTRIEVE/N	595	**455**
REPORT OPTION/N	345	**269**
C-TREE	395	**315**
R-TREE	295	**239**
C-TREE/R-TREE BUNDLE	650	**519**
DBC III	250	**169**
W/SOURCE CODE	500	**359**
DBC III PLUS	750	**595**
DBVISTA	195	**139**
FABS PLUS	195	**169**
INFORMIX ESQL/C	595	**CALL**
INFORMIX 4GL	995	**CALL**
INFORMIX SQL	795	**CALL**
PHACT MANAGER	249	**219**
EDITORS		
BRIEF	195	**CALL**
W/DBRIEF	275	**CALL**
CVUE W/SOURCE CODE	250	**199**
EDIX	195	**155**
EMACS	295	**265**
EPSILON	195	**149**
FIRSTIME (C)	295	**229**
KEDIT	125	**99**
PC/EDT	250	**229**
PC/VI	149	**109**
PI	195	**155**
PMATEPRO	195	**109**
SPF/PC	195	**145**
VEDIT PLUS	185	**129**
XTC	99	**79**
FORTRAN COMPILERS		
LAHEY FORTRAN	477	**CALL**
LAHEY PERSONAL FORTRAN 77	95	**89**
MICROSOFT FORTRAN	450	**269**
RM/FORTRAN	595	**399**
WATFOR FORTRAN	375	**335**
FORTRAN 77	350	**319**
FORTRAN UTILITIES/LIBRARIES		
ACS TIME SERIES	495	**389**
DIAGRAM'ER	129	**115**
DOCUMENT'ER	129	**115**
DIFF-E-Q	495	**445**
FORTLIB	125	**109**
FORTRAN ADDENDA	165	**139**
FORTRAN ADDENDUM	95	**85**
GRAFLIB	175	**169**
GRAFMATICS	135	**119**
MATHPAC	495	**445**
NO LIMIT	129	**115**
PLOTMATICS	135	**119**
SSP/PC	350	**269**
GRAPHICS		
ADVANTAGE GRAPHICS (C)	250	**225**

	LIST	OURS
ESSENTIAL GRAPHICS	250	**185**
GRAPHIC	350	**285**
GSS GRAPHIC		
DEVELOPMENT TOOLKIT	495	**375**
HALO	300	**205**
HALO (5 MICROSOFT LANG.)	595	**389**
HALO TEN FONT PACK	100	**79**
METAWINDOW	195	**159**
METAWINDOW PLUS	275	**229**
TURBOWINDOW/C	95	**79**
TURBO HALO (FOR TURBO C)	100	**79**
LINT		
PC-LINT	139	**99**
PRE-C	295	**155**
MODULA-2		
FARBWARE MODULA-2	90	**79**
LOGITECH MODULA-2		
APPRENTICE PACKAGE	99	**79**
WIZARD PACKAGE	199	**159**
MAGIC TOOLKIT	99	**79**
WINDOW PACKAGE	49	**39**
ROM PACKAGE AND CROSS		
RUNTIME DEBUGGER	299	**239**
REPERTOIRE	89	**75**
OPERATING SYSTEMS		
MICROPORT:		
286 DOS MERGE	149	**125**
SYSTEM V/AT (COMPLETE)	549	**465**
RUNTIME SYSTEM	199	**169**
SOFTWARE DEVELOPMENT SYSTEM	249	**209**
TEXT PREPARATION SYSTEM	199	**169**
UNLIMITED LICENSE KIT	249	**209**
SCO:		
XENIX SYSTEM V (COMPLETE)	1295	**995**
DEVELOPMENT SYSTEM	595	**499**
OPERATING SYSTEM	595	**499**
TEXT PROCESSING PACKAGE	195	**145**
WENDIN:		
OPERATING SYSTEM TOOLBOX	99	**79**
PCVMS	99	**79**
PCNX	99	**79**
WENDIN-DOS	99	**79**
WENDIN-DOS APPLICATION	99	**79**
PASCAL COMPILERS		
ADVANTAGE 386 PASCAL	895	**805**
MARSHAL PASCAL	189	**155**
MISCROSOFT PASCAL	300	**185**
PASCAL-2	350	**329**
PROFESSIONAL PASCAL	595	**CALL**
TURBO PASCAL NEW V. 4.0	100	**65**
TURBO PASCAL DEVELOPERS LIBRARY NEW	395	**259**
TURBO PASCAL ADD-ONS		
ALICE	95	**69**
DOS/BIOS & MOUSE TOOLS	75	**69**
FIRSTIME	75	**59**
FLASH-UP	89	**79**
FLASH-UP TOOLBOX	49	**45**
GSI PASCAL DEBUGGER	50	**45**
INSIDE TRACK	65	**55**
MACH 2	75	**59**
METRABYTE DATA ACQ. TOOLS	100	**89**
PASCAL HELPER	80	**75**
PEEKS & POKES	45	**39**
SCIENCE AND ENGIN. TOOLS	75	**69**
SCREENPLAY	175	**129**
SCREEN SCULPTOR	125	**95**
SYSTEM BUILDER	150	**129**
IMPEX	100	**89**
REPORT BUILDER	130	**115**
TP2C	249	**199**
TURBO•ASM	99	**69**
TDEBUGPLUS	60	**49**
TURBO ASYNCH PLUS	100	**79**
TURBO EDITOR TOOLBOX	70	**45**
TURBOEXTENDER	85	**65**
TURBO GRAPHIX TOOLBOX	70	**45**
TURBOHALO	129	**99**
TURBOMAGIC	99	**89**
TURBO MASTER	125	**99**
TURBO NUMERICAL METHODS	100	**65**

NEW PRODUCTS

PI EDITOR—Lightning fast editor with all the features of today's most popular programmer's editors and more. Among PI's 200+ commands and features are support for unlimited windows in any size, scrolling of windows simultaneously, background printing, and multiple file support. PI also has extensive language support for Assembly, Basic, C, and Pascal. Single stroke compile, and see your source and compilation errors simultaneously. Much more, and PI benchmarks faster than our leading selling editors.
List: $195 **Ours: $155**

RTC PLUS—Translator for converting either FORTRAN or RATFOR to C. No need to hand translate most of your code. RTC PLUS is perfect for translating FORTRAN libraries, and any FORTRAN code where I/O is concentrated in a few routines.
List: $450 **Ours: $399**

JYACC APPLICATION MANAGER (JAM)—Powerful application generator for creating, testing and modifying forms and windows. A unique application shell is created to create, and then specify control flow between the screens and windows.
List: $750 **Ours: $679**

THE AUTOMATED PROGRAMMER—Programming system for numerical computation that enables a computer to recognize and process mathematical, engineering, and scientific problem specs. Computational formulas are translated into FORTRAN. The AUTOMATED PROGRAMMER offers a range of output choices from structured reports to pictures. Computational results may be embedded in complex 2D text and graphic "IMAGE" formats.
List: $995 **Ours: $895**

TURBO OPTIMIZER	75	**65**
TURBO POWER TOOLS PLUS	100	**79**
TURBOPOWER UTILITIES	95	**79**
TURBO PROFESSIONAL	70	**49**
TURBOWINDOW/PASCAL	95	**79**
UNIVERSAL GRAPHICS LIBRARY	150	**119**
PROFILERS		
CODESIFTER	119	**85**
PFINISH	395	**209**
THE PROFILER	125	**89**
THE WATCHER	60	**55**
SCREEN DISPLAY/WINDOWS		
C POWER WINDOWS	130	**115**
C-SCAPE	279	**CALL**

	LIST	OURS
CURSES W/SOURCE CODE	250	169
GREENLEAF DATA WINDOWS	225	155
W/SOURCE CODE	395	269
JYACC FORMAKER	495	449
JYACC JAM	750	679
MICROSOFT WINDOWS	99	65
MS WINDOWS DEVELOPMENT KIT	500	309
PANEL	295	215
PANEL PLUS	495	395
PANEL/QC	129	95
PANEL/TC	129	95
QUICKSCREEN	195	175
SCREENACE	195	169
SCREENSTAR W/SOURCE	198	155
VIEW MANAGER	275	199
VITAMIN C	225	165
VC SCREEN	99	85
WINDOWS FOR C	245	189
WINDOWS FOR DATA	395	315
ZVIEW	245	169
TRANSLATORS		
BAS___C (ECONOMY)	199	169
BAS___C (COMMERCIAL)	375	319
BAS___PAS (ECONOMY)	149	125
BAS___PAS (COMMERCIAL)	280	239
BASTOC	495	399
BASTOC (BASICA VERSION)	795	639
DB2C	299	CALL
DBXTRANSLATOR	350	299
PTRAL	179	CALL
RTC PLUS	325	289
TP2C	249	199
TURBO TO C	495	449
ADDITIONAL LANGUAGES		
ACTOR	495	CALL
HS/FORTH	395	359
JANUS/ADA C PACK	95	89
LATTICE RPG II COMPILER	750	629
SCREEN DESIGN AID	350	309
SOURCE ENTRY UTILITY	250	199
SORT/MERGE UTILITY	250	199
MASTER FORTH	125	115
PC/FORTH	150	109
PERSONAL REXX	125	99
THE WEINER SHELL	199	179
ADDITIONAL PRODUCTS		
ADVANTAGE VCMS	379	329
DAN BRICKLIN'S DEMO PROGRAM	75	59
DAN BRICKLIN'S DEMO TUTORIAL	50	45
FAST FORWARD	70	59
FLOW CHARTING II	229	205
HELP/CONTROL	125	99
LINK & LOCATE	350	315
NORTON GUIDES	100	65
NS CHARTS	450	405
ON-LINE HELP	149	99
SAPIENS V8	300	269
SET: SCIL	349	319
SOFTSCREEN HELP	195	149

	LIST	OURS
SOURCE PRINT	95	75
TEXT MANAGEMENT UTILITIES	120	89
TREE DIAGRAMER	77	69
VENTURA PUBLISHER	895	625
BLAISE		
ASYNCH MANAGER	175	135
C TOOLS PLUS	175	135
EXEC PROGRAM CHAINER	95	75
LIGHT TOOLS FOR DATALIGHT C	100	79
PASCAL TOOLS	125	99
PASCAL TOOLS 2	100	79
PASCAL TOOLS & TOOLS 2	175	135
RUNOFF TEXT FORMATTER	50	45
TURBO ASYNCH PLUS	100	79
TURBO C TOOLS	129	99
TURBO POWER TOOLS PLUS	100	79
VIEW MANAGER	275	199
DIGITAL RESEARCH		
CB 86	600	539
PASCAL MT+	400	359
PL1___86	750	675
DR ASSEMBLER + TOOLS	200	179
ACCESS MANAGER	400	359
DISPLAY MANAGER	500	449
FORTRAN 77	350	319
LATTICE		
LATTICE C *SPECIAL PRICE*	500	CALL
W/SOURCE CODE	900	495
C CROSS REFERENCE GENERATOR	50	39
W/SOURCE CODE	200	139
C-FOOD SMORGASBORD	150	95
W/SOURCE CODE	300	179
C-SPRITE	175	119
CURSES SCREEN MANAGER	125	89
W/SOURCE CODE	250	169
CVUE	75	59
W/SOURCE CODE	250	199
DBC III	250	169
W/SOURCE CODE	500	359
DBC III PLUS	750	595
W/SOURCE CODE	1500	1185
LMK	195	139
LSE	125	99
RPG II DEVELOPER SYSTEM	1400	1119
RPG II COMPILER	750	629
RPG II SEU	250	199
RPG II SORT/MERGE	250	199
RPG II SCREEN DESIGN AID	350	309
SECRETDISK II	79	69
SIDETALK	120	89
SSP/PC	350	269
TEXT MANAGEMENT UTILITIES	120	89
LIFEBOAT		
ADVANTAGE 386 C	895	805
ADVANTAGE 386 PASCAL	895	805
ADVANTAGE C++	495	479
ADVANTAGE DISASSEMBLER	295	265
ADVANTAGE GRAPHICS	250	225
ADVANTAGE LINK	395	359

	LIST	OURS
ADVANTAGE MAKE	125	99
ADVANTAGE VCMS	379	329
PANEL	295	215
PANEL PLUS	495	395
QUICKSCREEN	195	175
RUN/C – THE C INTERPRETER	120	79
RUN/C PROFESSIONAL	250	155
TIMESLICER	295	265
W/SOURCE CODE	1000	895
MICROSOFT		
MS BASIC COMPILER (XENIX)	695	419
MS BASIC INTERPRETER (XENIX)	350	209
MS C COMPILER	450	269
MS COBOL COMPILER	700	439
FOR XENIX	995	609
MS FORTRAN	450	269
FOR XENIX	695	419
MS LEARNING DOS	50	39
MS LISP	250	155
MS MACRO ASSEMBLER	150	95
MS MOUSE BUS VERSION	175	119
MS MOUSE SERIAL VERSION	195	125
MS MUMATH	300	185
MS PASCAL COMPILER	300	185
FOR XENIX	695	419
MS QUICK BASIC	99	65
MS QUICK C	99	65
MS SORT	195	125
MS WINDOWS	99	65
MS WINDOWS DEVELOPMENT KIT	500	309
PHOENIX		
PASM86	195	109
PDISK	145	99
PFANTASY PAC	995	599
PFINISH	395	209
PFIX86PLUS	395	209
PFORCE	395	209
PFORCE++	395	209
PLINK86PLUS	495	275
PLOCATE	CALL	CALL
PMAKER	125	79
PMATEPRO	195	109
PRE-C	295	155
PTEL	195	109
POLYTRON		
POLYBOOST	80	69
POLYDESK III	99	75
POLYDESK III ARCHIVIST	50	45
POLYDESK III CRYPTOGRAPHER	50	45
POLYDESK III TALK	70	55
POLYLIBRARIAN	99	85
POLYLIBRARIAN II	149	125
POLYMAKE	149	125
POLYSHELL	149	125
POLYTRON C BEAUTIFIER	49	45
POLYTRON C LIBRARY I	99	75
POLYXREF	219	185
POLYXREF ONE LANGUAGE	129	109
PVCS NETWORK	1000	CALL

	LIST	OURS
PVCS CORPORATE	395	329
PVCS PERSONAL	149	125
PVMFM	199	155
BORLAND		
EUREKA EQUATION SOLVER	100	65
REFLEX & REFLEX WORKSHOP	200	129
REFLEX DATA BASE SYSTEMS	150	89
REFLEX WORKSHOP	70	45
SIDEKICK & TRAVELING SIDEKICK	125	85
SIDEKICK	85	59
TRAVELING SIDEKICK	70	45
SUPERKEY	100	65
TURBO BASIC COMPILER	100	65
TURBO DATABASE TOOLBOX	100	65
TURBO EDITOR TOOLBOX	100	65
TURBO TELECOM TOOLBOX	100	65
TURBO C COMPILER	100	65
TURBO JUMBO PACK	300	219
TURBO LIGHTNING	100	65
TURBO PASCAL NEW V. 4.0	100	65
TURBO PASCAL DEV LIB NEW	395	259
TURBO PASCAL AND TUTOR	125	85
TURBO TUTOR	40	29
TURBO PASCAL NUMERICAL METHODS TOOLBOX	100	65
TURBO PASCAL DATABASE TOOLBOX	70	45
TURBO PASCAL EDITOR TOOLBOX	70	45
TURBO PASCAL GAMEWORKS TOOLBOX	70	45
TURBO PASCAL GRAPHICS TOOLBOX	70	45
TURBO PROLOG COMPILER	100	65

XENIX/UNIX PRODUCTS

ADVANTAGE C++	695	CALL
BTRIEVE	595	455
C-TERP	498	379
INFORMIX	CALL	CALL
KORN SHELL	125	115
MICROSOFT LANGUAGES	CALL	CALL
PANEL PLUS	795	675
REAL TOOLS	149	89
RM/COBOL	1250	949
RM/FORTRAN	750	549
TURBO PROLOG TOOLBOX	100	65
WORD WIZARD	70	49
WORD WIZARD AND TURBO LIGHTNING	150	95
AMIGA PRODUCTS		
AZTEC COMMERCIAL	499	449
AZTEC DEVELOPERS	299	269
DBMAN	150	119
LATTICE AMIGA DOS COMPILER	225	179
LATTICE PRO AMIGA C COMPILER	375	299
MODULA II REG	90	75
MODULA II DEV	150	125
TRUE BASIC	100	79

ADVANTAGE C++™

Add ADVANTAGE C++ to your C compiler and enhance your existing C tools with new object-oriented capabilities. The latest version, **now available with MS Windows Support.** is the *only full implementation* of AT&T's C++ language. This superset of C gives you the speed, support and reliability you need to develop large and complex programs with greater resilience, fewer bugs. **List $495 Ours $479**

LIFEBOAT

Periscope

Debug device drivers, keyboard handlers, memory-resident programs, interrupt driven-programs, non-DOS programs, even DOS. Use on IBM PC/XT/AT and close compatibles with most any language. See source and/or symbols from your programs. Four models to choose from—two have write-protected RAMBOARDS to protect the periscope software and free up DOS memory for your programs. See August 1987 PC Tech Journal Review; August 17th InfoWorld User Profile. **List $345 Ours $289**

PERISCOPE

Greenleaf C Sampler

3-in-1 oil for your C programs. Interrupt communications, windows, menus and more in a big new library. Device independent, logical attributes, unlimited logical windows. Keyboard input includes function and other keys. Clear examples to help you get started. Drives comm ports up to 9600 baud, with XMODEM included, using the most powerful system available. Also includes pull down menus that are easy to use. Available for Quick C and Turbo C. Free source code with purchase by Nov. 15th. **List $95 Ours $79**

GREENLEAF Software

Microsoft C v. 5.0

Fast code, fast compilation, fast prototyping and fast developing makes Microsoft C 5.0 the ultimate C development environment. Includes optimizing compiler for fast execution speed, QuickC for fast compilation and prototyping and CodeView for fast debugging. New version includes optimizations such as Inline code generation and constant folding new graphics library, more comprehensive ANSI features, extensive math libraries, full memory model support, faster linker, enhanced error reporting and many more improvements. **List $450 Ours $269**

Microsoft

SCREEN ACE Form Master

The Screen-Ace Form Master is designed to create and maintain forms, tables, text screens, and menus. Form Master is flexible. You deal with whole forms instead of individual characters, leaving you with more time to spend on the application. New features in the latest release include mouse support, keyboard remapping and redefining, pop-up windows, data validation and graphics mode support. Form Master is coded entirely in Assembler, so its Fast! **Specify language interface. List $195 Ours $169**

ace SOFTWARE PRODUCTS, INC.

PforCe

PforCe is a pre-coded optimized object-oriented toolkit of over 400 routines for C programmers. It includes data bases with B-trees, windows, interrupt-driven communications, string handling, menus, all of the basic DOS interfaces, and a complete set of low-level functions to interface directly to the hardware. PforCe comes complete with indexed reference manual, on line resident help, and quick reference card. **List $295 Ours $209**

Phoenix

Terms and Policies
- **We honor MC, VISA, AMERICAN EXPRESS**

No surcharge on credit card or C.O.D. Prepayment by check. New York State residents add applicable sales tax. Shipping and handling $3.00 per item, sent UPS ground. Rush service available, prevailing rates.
- Programmer's Paradise will match any current nationally advertised price for the products listed in this ad.
- Prices and Policies subject to change without notice.
- ***New* Extended Hours 9AM EST—7PM EST**

*Ask for details. Some manufacturers will not allow returns once disk seals are broken.

Corporate Buyers—Call for special discounts and benefits!

1-800-445-7899
In NY: 914-332-4548
Customer Service:
914-332-0869
International Orders:
914-332-4548
Telex: 510-601-7602

Programmer's Paradise™

A Division of Hudson Technologies, Inc.
42 River Street, Tarrytown, NY 10591

CIRCLE 32 ON READER SERVICE CARD

C CODE FOR THE PC

source code, of course

C Source Code

Bluestreak Plus Communications (two ports, programmer's interface, terminal emulation)	$400
Greenleaf Data Windows (windows, menus, data entry, interactive form design)	$315
Barcode Generator (specify Code 39 (alphanumeric), Interleaved 2 of 5 (numeric), or UPC)	$300
GraphiC 4.0 (high-resolution, DISSPLA-style scientific plots in color & hardcopy)	$275
Vitamin C (MacWindows)	$200
Greenleaf Communications Library (interrupt mode, modem control, XON-XOFF)	$160
Greenleaf Functions (296 useful C functions, all DOS services)	$160
Essential C Utility Library (400 useful C functions)	$160
Essential Communications Library (C functions for RS-232-based communication systems)	$160
PC/IP (CMU/MIT TCP/IP implementation for PCs)	$100
B-Tree Library & ISAM Driver (file system utilities by Softfocus)	$100
The Profiler (program execution profile tool)	$100
Entelekon C Function Library (screen, graphics, keyboard, string, printer, etc.)	$100
Entelekon Power Windows (menus, overlays, messages, alarms, file handling, etc.)	$100
QC88 C compiler (ASM output, small model, no longs, floats or bit fields, 80+ function library)	$90
CBTree (B+tree ISAM driver, multiple variable-length keys)	$80
ME (programmer's editor with C-like macro language by Magma Software)	$75
Wendin PCNX Operating System Shell	$75
Wendin PCVMS Operating System Shell	$75
Wendin Operating System Construction Kit	$75
EZ_ASM (assembly language macros bridging C and MASM)	$60
Multi-User BBS (chat, mail, menus, sysop displays; uses Galacticomm modem card)	$50
Heap Expander (dynamic memory manager for expanded memory)	$50
Make (macros, all languages, built-in rules)	$50
Vector-to-Raster Conversion (stroke letters & Tektronix 4010 codes to bitmaps)	$50
Coder's Prolog (inference engine for use with C programs)	$45
PC/MPX (light-weight process manager; includes preemption and cooroutine packages)	$45
Biggerstaff's System Tools (multi-tasking window manager kit)	$40
TELE Kernel (Ken Berry's multi-tasking kernel)	$30
TELE Windows (Ken Berry's window package)	$30
Clisp (Lisp interpreter with extensive internals documentation)	$30
Translate Rules to C (YACC-like function generator for rule-based systems)	$30
6-Pack of Editors (six public domain editors for use, study & hacking)	$30
ICON (string and list processing language, Version 6 and update)	$25
LEX (lexical analyzer generator)	$25
Bison & PREP (YACC workalike parser generator & attribute grammar preprocessor)	$25
C Compiler Torture Test (checks a C compiler against K & R)	$20
PKG (task-to-task protocol package)	$20
A68 (68000 cross-assembler)	$20
Small-C (C subset compiler for 8080 and 8088)	$20
tiny-c (C subsubset interpreter including the tiny-c shell)	$20
Xlisp 1.5a (Lisp interpreter including tiny-Prolog in Lisp)	$20
List-Pac (C functions for lists, stacks, and queues)	$20
XLT Macro Processor (general purpose text translator)	$20
C Tools (exception macros, wc, pp, roff, grep, printf, hash, declare, banner, Pascal-to-C)	$15

Data

DNA Sequences (GenBank 48.0 of 10,913 sequences with fast similarity search program)	$150
Protein Sequences (5,415 sequences, 1,302,966 residuals, with similarity search program)	$60
Webster's Second Dictionary (234,932 words)	$60
U. S. Cities (names & longitude/latitude of 32,000 U.S. cities and 6,000 state boundary points)	$35
The World Digitized (100,000 longitude/latitude of world country boundaries)	$30
KST Fonts (13,200 characters in 139 mixed fonts: specify TEX or bitmap format)	$30
NBS Hershey Fonts (1,377 stroke characters in 14 fonts)	$15
U. S. Map (15,701 points of state boundaries)	$15

The Austin Code Works
11100 Leafwood Lane
Austin, Texas USA 78750-3409

Voice: (512) 258-0785
BBS: (512) 258-8891
Email: FidoNet 1:382/12

Free surface shipping on prepaid orders

MasterCard/VISA

CIRCLE 33 ON READER SERVICE CARD

PUBLIC DOMAIN SOFTWARE REVIEW

Pascal programming aids and PC-FASTYPE

By Tim Parker

For most programmers, the hardest part of coding is debugging. When this stage arrives, programmers turn to a number of utilities. A source listing formatter, variable cross-referencer, and syntax checker are becoming standard tools for all languages. They dramatically reduce debugging time and make it much less of an onerous task.

Useful utilities are available in the commercial market and the public domain. Almost every language has a good package available, usually written by a programmer who spends most of his or her development time in that language. Some programmers become dependent on these utilities since they are superb products and excellent productivity aids.

Although the public domain has always had formatters, variable cross-referencers, and other utilities, most have been simple, rudimentary programs. But with the development of improved compilers, debuggers became more powerful; and as the use of languages became more widespread, the demand for commercial products arose.

In the September column, I discussed a utility program called SNAP! for the dBASE family ("dBASE becomes a SNAP!; a C graphics library," *COMPUTER LANGUAGE,* Sept. 1987, pp. 77-79). SNAP! matched commercial products in every regard and exceeded most. I do virtually all of my code checking in dBASE with SNAP!, and I highly recommend it for any dBASE programmer.

Thanks to the evolution of low-cost, full-feature compilers, Pascal has really caught on over the last few years, resulting in a number of new tools for this language. From the early days of CP/M, when JRT Pascal became available for about $50, to the emergence of Borland International's Turbo Pascal, Pascal has become the most popular language to learn first, replacing BASIC for IBM PCs and similar machines. Borland International's compiler, now heading into its fourth edition, still costs under $100.

With the popularity of Turbo Pascal, a new market of commercial products written in the language has arisen. Most of these are extensions and utilities. The public domain has benefitted from the large number of Turbo Pascal users, too. I have reviewed many Turbo Pascal utilities, such as Boosters, a group of machine language functions that enhance the language ("Libraries for C—and one for Turbo Pascal," *COMPUTER LANGUAGE,* Feb. 1987, pp. 91-94).

However, despite casual mentions of programs encountered on SIG disks, I haven't focused on a good productivity aid like SNAP! for Turbo Pascal because I could never find one that was good enough. Oh, there were formatters and a few cross-referencers, but none of them could be called anything more than utilities.

T-Ref parses and produces formatted code and variable cross-referencing for compilable Turbo Pascal code

However, this situation has changed, and a useful program that will be of interest to all Turbo Pascal users has finally crossed my desk.

T-Ref is a combined source lister and cross-referencer in Pascal that deserves a look. T-Ref requires DOS 2.0 or later and a minimum of 256K RAM, although at least 512K RAM is highly recommended. T-Ref works with Turbo Pascal versions 2 and 3. The program parsed just about any Pascal code that didn't include too many deviations from Turbo Pascal's command set.

T-Ref imposes a few limitations on code and variable checking. The first 20 characters of a variable are used for identification and the rest are ignored. The maximum string length is 130 characters and the maximum column width 132 characters. Depth-level checking is a maximum of 20 levels.

In checking code, T-Ref has one major flaw I suspect will be corrected with a future enhancement: it doesn't recover well from errors. When T-Ref is run on source code containing errors, such as mismatched conditionals or unrecognized symbols, it finds the error and bombs out, producing a cross-reference list of variables to that point and closing the formatted file output, if either or both are requested. An error message is then displayed.

T-Ref does not claim to be a syntax checker. However, most programmers would probably prefer a package that scans code and generates a list of errors, enabling them to debug faster. Instead, T-Ref should be run only on code that will compile (the manual points this out). A proper syntax checker would greatly improve T-Ref's usefulness.

T-Ref parses and produces formatted code and variable cross-referencing for compilable Turbo Pascal code. The program handles valid declaration sections of source code, from constants to procedures and functions. This feature allows a library of procedures or functions to be parsed for future reference.

When executed, T-Ref presents a full screen display used as the main menu. The top part of the screen indicates the current active directory, input file, and output file. A number of option toggles are indicated below that in two areas of the screen. One area refers to source listing options, the other to cross-referencing. The bottom section of the screen contains messages and key prompts.

Pop-up menus are available with T-Ref. The F1 help function key produces a list of what the function keys do, while the F2 function key is context-sensitive, producing relevant help information depending on the cursor location. Other pop-up menus are used for system options (such as header information for the source listing), cross referencing, and other parameters. A configuration can be saved as the default type with a function key.

The source listing parameters control a number of options. Reserve words can be formatted into upper case or forced to lower case or into formal naming conventions. (Formal naming usually refers to capitalizing the first letter of the reserved word, with the subsequent letters in lower case.) Similar formatting of identifiers can also be controlled.

The source listing can have line numbers printed in the left margin. If files are *included*, the line numbering splits into two, with one column providing numbering for the entire file and the line number held constant while the *included* file is produced, and the second column providing numbering for the particular *included* file. When source code is listed and files are *included*, the source *includes* the full source code for that *included* file also. This option can be turned off to prevent parsing of *included* material.

An option called lexical level produces a listing with the depth of conditional or other tiered code indicated by a digit. Thus, a procedure is numbered as 1 when printed until it calls another procedure, then the number increases to show the level is now 2. As the levels are closed, the numbering decreases.

Block levels can be numbered in the same manner. When a *BEGIN, FOR,* or similar statement is encountered, the numbering increases; as *END* statements are encountered, the numbering decreases. An active procedure column on the right of the listing produces in brackets the name of the function, procedure, or program active at that point.

This numbering creates a fairly strange-looking listing, especially if all the numbering options are active. Therefore, line numbers can be in the left column, increasing steadily unless an *include* statement is encountered, where it holds the number steady and a second column appears that increases through the *included* code. If levels and blocks are active, two additional number columns appear. Finally, the active control code is presented on the right. These options are useful for debugging purposes (although T-Ref isn't designed for debugging since it crashes on bugs) and even better for documentation and determining logic flow.

If a header field is desired, it can be printed at the top of each source page. Headers have a maximum of five lines and can be up to 75 characters wide. Page numbering can also be toggled, with a date and time mark printed.

The cross-reference section produces a list of the variables in the code and information about the type of variables. All occurrences of a particular variable are noted.

The format of the cross-reference listing is different from most others. Next to the name of the variable, the basic type (label, constant, function, etc.) and the data type (integer, byte, Boolean, etc.) are noted. The declaration's line number appears on the next line (along with procedure name and number, if applicable) with the lexical level of the declaration within the overall program. If the declaration is not encountered within the examined code (for example, when a library that uses a global variable is parsed), a statement indicating this is printed. Each occurrence of the variable is then noted on subsequent lines with the line number.

Reserved words are identified by T-Ref. Reserved Turbo Pascal words for Turbo versions to 3.01 are included in the utility I tested and listed in an appendix of the manual. An asterisk indicates that the variable was explicitly set on the occurrence.

T-Ref checks procedures and functions for recursion and notes the results in the printout. Formal parameters explicitly set in the code are also noted in the

Technical Writer

Software Documentation

If you're looking for a company where every day is filled with challenge, you've found it. If you can combine technical writing skills with creativity, the ability to manage projects, and a can-do attitude, Microsoft is interested in you. Microsoft is looking for experienced Technical Writers to join our award-winning documentation teams.

The ideal candidates will have the following qualifications:

- Writing ability. 2-5 years' proven success in a technical field, preferably in a production environment in the microcomputer industry.
- Software knowledge. Knowledge of and/or experience with applications products and languages for MS-DOS and MAC environments; experience with graphics and windows; knowledge of MS-DOS; programming experience preferred (some positions require demonstrated ability to write "C" programming).
- Project management. Strong organizational and communications skills, plus the ability to manage a project to meet deadlines in a dynamic environment.

Microsoft offers an excellent benefits package, and is located in Seattle, the heart of the beautiful Northwest. If you can rise to the challenge and are ready to work in an environment that values teamwork, yet rewards independence, we'd like to hear from you.

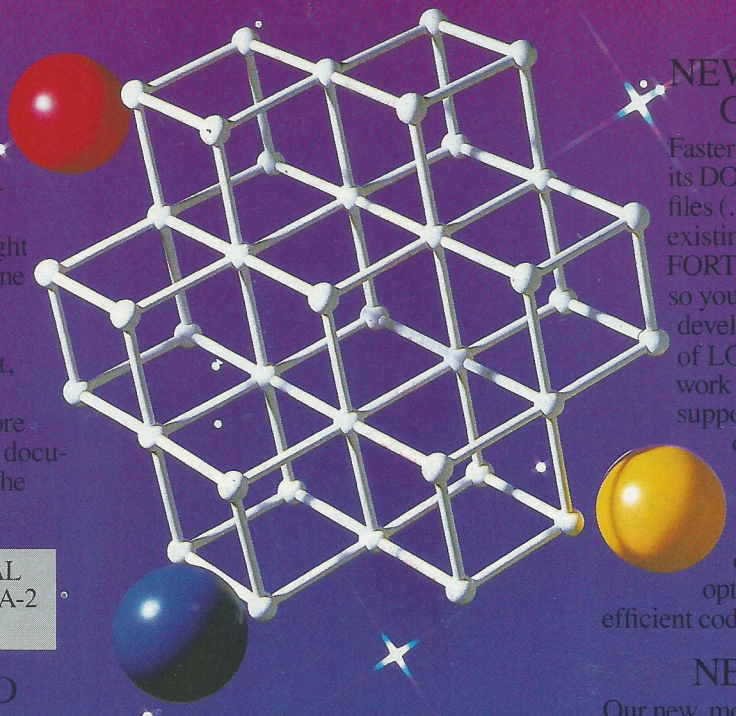
For consideration, please forward your resume and samples of work you've written, along with salary history to: MICROSOFT CORPORATION, Human Resources, Dept. JRWR-11CL, 16011 N.E. 36th Way, P.O. Box 97017, Redmond, WA 98073-9717. We are an equal opportunity employer. Samples returned only upon request.

Microsoft®

SOFTWARE ENGINEERING COMES OF AGE.

ANNOUNCING LOGITECH MODULA-2 VERSION 3.0

Modula-2 is the language of choice for modern software engineering, and LOGITECH Modula-2 is the most powerful implementation available for the PC. The right language and the right tools have come together in one superior product. Whether you're working on a small program or a complex project, with LOGITECH Modula-2 Version 3.0 you can write more reliable, maintainable, better documented code in a fraction of the time at a fraction of the cost.



FREE TURBO PASCAL TO LOGITECH MODULA-2 TRANSLATOR

NEW, IMPROVED COMPILER

Faster and more flexible. Now its DOS linker compatible object files (.OBJ) can be linked with existing libraries in C, PASCAL, FORTRAN and ASSEMBLER — so you can build on previous development and put the power of LOGITECH Modula-2 to work for you right now. Fully supports Wirth's latest language definition, including LONGINT and LONGSET, which provides large set support including SET of CHAR. Provides optimization for tighter, more efficient code generation.

NEW, IMPROVED DEBUGGERS

Time gained with a fast compiler can be lost at debug time without the right debugging tools. With the powerful Logitech Modula-2 Debuggers you can debug your code *fast*, and dramatically improve your overall project throughput. The Post Mortem Debugger analyzes the status of a program after it has terminated while the dynamic, Run Time Debugger monitors the execution of a program with user-defined breakpoints. With their new, mouse based, multiple-window user interface these powerful debugging tools are a pleasure to use.

NEW, INTELLIGENT LINKER

Links only those routines from a particular module that you need, so you eliminate unreferenced routines and produce smaller, more compact executable files.

NEW EDITOR

Our new, mouse based editor is fully integrated, easy to learn, fast and easy to use, and very customizable. Its multiple, overlapping windows and color support make it easy to manage parts of one file or several files on the screen at one time. You'll love using it — with or without a mouse.

Call for information about our VAX/VMS version, Site License, University Discounts, Dealer & Distributor pricing.

To place an order call toll-free:
800-231-7717
In California:
800-552-8885

☐ LOGITECH Modula-2 V. 3.0 Compiler Pack — **$99**
Compiler in overlay and fully linked form, Linkable Library, Post Mortem Debugger, Point Editor

☐ LOGITECH Modula-2 V. 3.0 Toolkit — **$169**
Library sources, Linker, Run Time Debugger, MAKE, Decoder, Version, XRef, Formatter

☐ LOGITECH Modula-2 V. 3.0 Development System — **$249**
Compiler Pack plus Toolkit

☐ Turbo Pascal to Modula-2 Translator — **FREE**
With Compiler Pack or Development System

☐ Window Package — **$49**
Build true windowing into your Modula-2 code.

☐ Upgrade Package
Call LOGITECH for information or to receive an order form.

Add $6.50 for shipping and handling. California residents add applicable sales tax. Prices valid in U.S. only. Total Enclosed $ _____

☐ VISA ☐ MasterCard ☐ Check Enclosed

Card Number _____ Expiration Date _____

Signature _____

Name _____

Address _____

City _____ State _____

Zip _____ Phone _____

LOGITECH

LOGITECH, Inc.
6505 Kaiser Drive, Fremont, CA 94555
Tel: 415-795-8500
In Europe: LOGITECH, Switzerland
Tel: 41-21-87-9656 Telex 458 217 Tech Ch
In the United Kingdom: LOGITECH, U.K.
Tel: 44908-368071 Fax: 44908-71751

Turbo Pascal is a registered trademark of Borland International. VAX and VMS are registered trademarks of Digital Equipment Corp.

CIRCLE 35 ON READER SERVICE CARD

printout and don't cause an error. (There are times when explicit assignment of formal parameters are useful.)

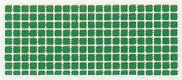
T-Ref also produces a procedure and function map, listing the procedures and functions that have been parsed and the declaration line number. The total number of lines parsed (identifier type by basic type and data type) and the total number of identifiers are given.

Headers and page formatting options similar to those I described for the source printout are available within the cross-reference section. These options are not necessarily the same as those of the formatter, giving the user some control over printing headers in the source but not in the variable cross-reference (or a similar arrangement).

The manual accompanying T-Ref is about 30 pages long and provides a good discussion of the program and its capabilities. The options are discussed and examples are given. A listing of error messages T-Ref generates is included in an appendix.

T-Ref performed exactly as it should have. I tested it with Roy MacLean's FICALCS, a program I discussed in the January issue ("Financial calculations and Batch files, *COMPUTER LANGUAGE*, Jan. 87, pp. 83-85). T-Ref handled FICALCS easily despite FICALC's size and complexity and produced useful source code printouts. I found the variable cross-reference a bit confusing to read, but I'm used to the simpler cross-reference listings that provide the variable name, type, and all occurrences in a few lines. The verbosity of T-Ref's printout could be useful, although cross-referencing after the fact is usually performed for modification and documentation only.

If a code syntax checker and subsequent error display were added to T-Ref, users would be able to cross check and format source as they develop code instead of when they complete it. However, for Turbo Pascal programmers, T-Ref is a useful utility and well worth acquiring. Although I worked with v. 1.01, v. 1.1 is now available with a number of additional functions. A donation of $35 is requested to become a registered user of T-Ref and receive the latest updates.



PC-FASTYPE is a shareware program designed specifically for IBM PCs and compatibles. The author requests a $20 donation if it is found useful. Two versions of the program are distributed: one for the older keyboards and one for the enhanced keyboards. PC-FASTYPE requires a CGA or similar (EGA, Compaq, etc.) card to work. It will not function on a Hercules card.

PC-FASTYPE performs several tasks, including presenting a series of drills to practice and tie the user's keyboard pounding. It also has lessons for touch-typing on IBM keyboards. (Those with modified, nonstandard keyboards or IBM-compatible keyboards may have to adjust.)

PC-FASTYPE makes extensive use of the function keys and has numerous menus. A highlight system is used for menu picks. The program is well presented, and a computer novice can easily use it. PC-FASTYPE is fast and easily handled my 85 words-per-minute test. It was originally written in BASICA but was converted and compiled with QuickBASIC.

PC-FASTYPE can be set for beginner, intermediate, and expert levels, and the difficulty of the lessons and tests accelerates as the level increases. When in testing mode, PC-FASTYPE presents a text box after you've typed that gives a number of interesting parameters, such as a true words-per-minute rate (considering errors), raw words-per-minute rate (ignoring errors), the number of keystrokes, etc.

The documentation comes in two files: one contains the manual and the other an update for the new version (I tested v. 3.0). The documentation is well written and clearly explains how to use the program.

In testing, PC-FASTYPE presented lessons, measured typing speed, and provided practice in pounding the keys. It locked up my machine a couple of times in early tests because of memory residents.

Not everyone needs a program like PC-FASTYPE, but for those who want to learn touch-typing, improve their typing skills, or determine their typing speed, it is well suited. The limitation of CGA or EGA is unfortunate, and a Hercules version would probably be popular.

Tim Parker is a systems programmer in Ontario, Canada, who also writes for several technical magazines.

Products mentioned

T-Ref
Synergy Software of Nebraska
904 Lariat Cir.
Papillion, Neb. 68046

PC-FASTYPE
Trendtech
14 Ella Ln.
Wayne, N.J. 07470
(201) 694-8622

SNAP!
Walter Kennamer
1801 E. 12th St.
Apt. 1118
Cleveland, Ohio 44114
(216) 861-3646

TRUE MULTITASKING

With

MultiDos Plus

"multitasking for the IBM-PC."

Ideal for developing applications in process control, data acquisition, communications, and other areas. Check these features which make **MultiDos Plus** an unbeatable value.

- Run up to 32 programs concurrently.
- Your software continues to run under DOS. No need to learn a new operating system.
- Use the compilers you already have. Supports software written in most languages.
- Operator commands to load/run programs, change priority, check program status, abort/suspend/resume programs.
- Programmatic interface via INT 15H for the following.
  - * Intertask message communication. Send/receive/check message present on 64 message queues.
  - * Task control by means of semaphores. Get/release/check semaphores.
  - * Change priority-256 priority levels.
  - * Suspend task for specified interval.
  - * Spawn and terminate external and internal tasks.
  - * Disable/enable multitasking.
  - * and more!
- Independent foreground/background displays.
- Access to DOS while applications are running.

Hardware/Software Requirements

IBM PC/XT/AT or true clone. Enough memory to hold **MultiDos Plus** (48 KB) and all your application programs. Also may need 4 or 16 KB memory for "hidden screens" for each active task. MS-DOS (or PC-DOS) 2.0 or later operating system.

only: $24.95 *OR* $99.95 with source code

Outside USA add $5.00 shipping and handling.
Visa and Mastercard orders only call toll-free: 1-800-872-4566, ext. 350., or send check or money order to:

NANOSOFT
13 Westfield Rd, Natick, MA 01760
MA orders add 5% sales tax.

CIRCLE 36 ON READER SERVICE CARD

Generics
Software
Limited

AnimAID

DISPLAY THE POWER OF ADA™

AnimAID lets you build better operator efficiency into your Ada application through the use of sophisticated graphical interfacing techniques. Combining object-oriented applications building blocks with a graphical interface generation tool, AnimAID brings a wealth of ingenuity to your Ada applications.

AnimAID facilitates the tailorability of the applications interface to ever changing end user requirements. Here are some of the benefits of using AnimAID.

- Configurable graphics user interface
- Standardisation of development method
- Reusability of applications modules
- Highly reliable package generation
- Rapid prototyping of graphical interfaces
- Graphic-oriented applications prototyping
- Maximum use of graphics with no direct graphics programming

We'd like to tell you more about AnimAID. Please call us.

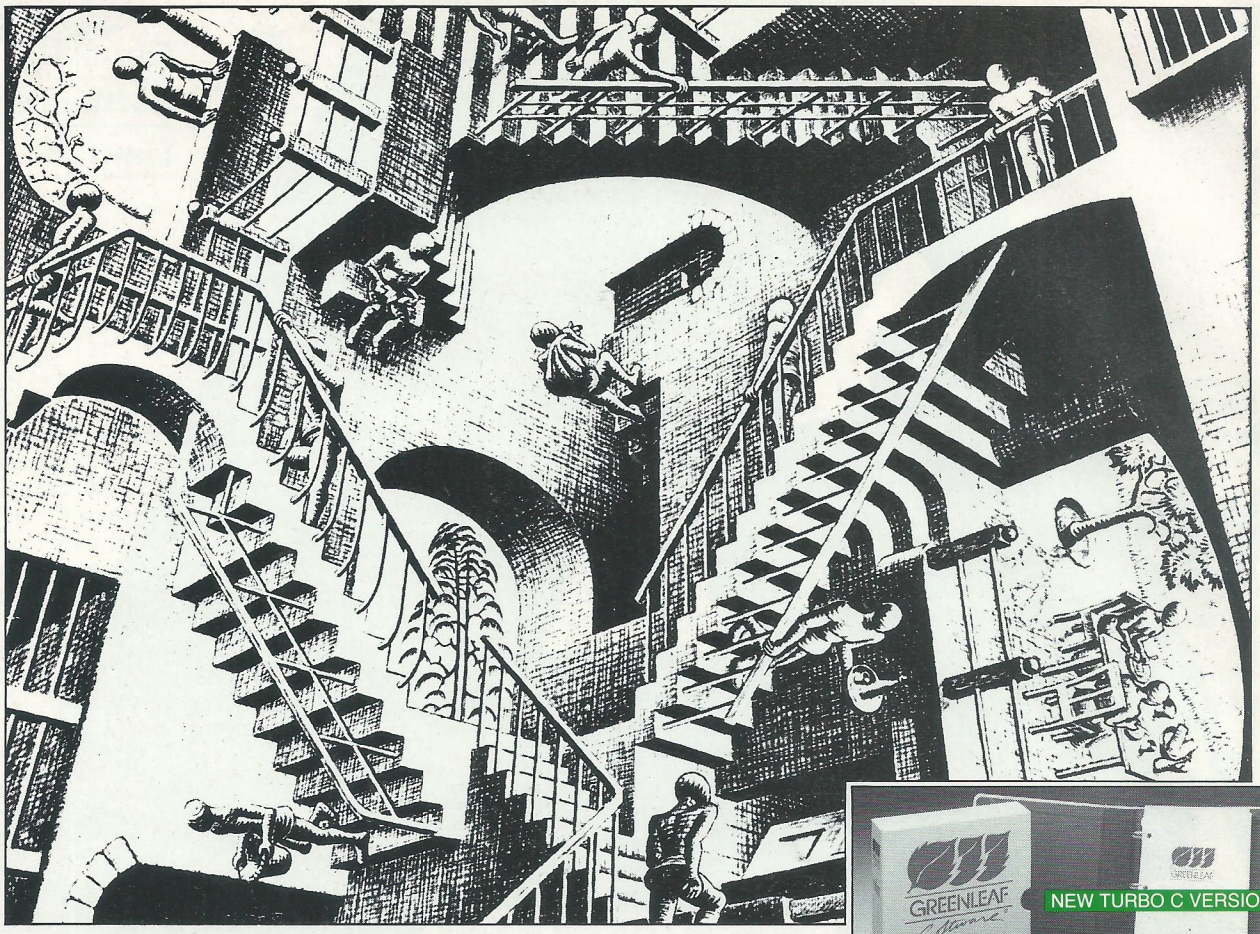
Generics Software Limited
7 Leopardstown Office Park
Foxrock, Dublin 18
Ireland

Telephone +353·1·954012
Telex 265871 MONREF G
Quoting ref 74:EIM229
Fax +353·1·841990

GENERICS

Ada is a registered trademark of the U.S.Government (AJPO)
Generics and AnimAID are trademarks of Generics Software Limited

CIRCLE 37 ON READER SERVICE CARD



Avoid extra steps.

You've got better things to do than repeat the same steps. Over . . . and over . . . and over. Up your productivity with Greenleaf Software.

With more than 70 new functions added to our popular libraries, Greenleaf is now the most complete and mature C language function resource available. It's no wonder we've been rated the best. Winning program developers in major corporations such as IBM, EDS and GM have proven our reliability in thousands of applications.

Step Lively

New Greenleaf Functions v.3.10 includes 295 of the functions you've been asking for — DOS, disk, video, color text and graphics, string, time/date, keyboard, plus many more! With Greenleaf, you'll finish faster.

Cut Corners

When it comes to merging information, the new Greenleaf Comm Library v.2.10 is the fastest communications facility of its kind. Over 120 functions — ring buffered, interrupt-driven asynchronous communications. And, only Greenleaf gives you the power to build a 16-port communication system.

Get on the Fast Track

Order your new Greenleaf library today! See your dealer or call 1-800-523-9830.

Greenleaf Comm Library	$185.00
Greenleaf Functions	$185.00
Greenleaf DataWindows	$225.00
Greenleaf C Sampler	$ 94.50
Digiboard Comm4	$325.00
Digiboard Comm8	$535.00

In stock, shipped next day.

Greenleaf DataWindows and Turbo C

DataWindows, the finest C programming windows tool available, puts windows, transaction data entry and menus at your fingertips.

Our new TURBO C versions are ready to get you going fast! And, our new 3-in-1 C Sampler for only $94.50 supports both Turbo C and Quick C with comm, windows, menus and more! Our libraries support all popular C compilers for MS DOS.

GREENLEAF Software

Call Toll Free:
800-523-9830

In Texas and Alaska:
214-446-8641

Greenleaf Software, Inc.
16479 Dallas Parkway, Suite 570
Dallas, Texas 75248

CIRCLE 38 ON READER SERVICE CARD

EXOTIC LANGUAGE OF THE MONTH CLUB

Beyond text editors with VAXTPU

By Leonard Struttmann

The VAX Text Processing Utility (VAXTPU) has been used to create remarkable text editors, but its usefulness extends far beyond that. This article introduces the elements of VAXTPU's powerful, structured programming language and gives you a set of sample procedures that illustrates how VAXTPU can also be a simple yet powerful data base report generator.

VAXTPU is used to develop text-processing interfaces. That doesn't sound very exciting until you realize that a text-processing interface may be anything from a customized text editor for a specific application to something as sophisticated as a data base manager. This diversity is achieved through a powerful structured programming language that includes a rich set of built-in procedures.

Theoretically, you could use VAXTPU to create an entirely new text-processing interface from scratch. The result would be your own customized text editor. It would also be a waste of time since VAXTPU comes with two interfaces already programmed in VAXTPU. One emulates the venerable EDT editor. The other implements the Extensible VAX Editor (EVE), an exceptionally easy yet powerful full-screen editor. EVE's features include multiple open files and text windows.

Either EDT or EVE can be customized or extended to meet your interface needs. I will illustrate the basics you need to use the rich VAXTPU programming language to extend the EVE interface and include your very own text-processing functions.

One of VAXTPU's unique features is that it has no explicit statements to declare the data types of variables. In fact, variables may have any of VAXTPU's data types, and can even change the data types in midprogram.

The data type of a variable is set when that variable is used on the left-hand (result) side of an assignment statement. It is set to the type that results from the expression on the right-hand side of the assignment. For example, the following assignment not only assigns the value of 12346 to the variable *b_count*, but it also defines the variable *b_count* to be of type *INTEGER:*

```
b_count := 12346;
```

With this freedom in defining data types comes a responsibility for the programmer because VAXTPU's built-in procedures expect and return parameters of specific data types. The data types used by VAXTPU include:

- *INTEGER*: Whole numbers from -2147483648 to +2147483647.
- *STRING*: Character data. A string may be from 0 to 65,535 characters long.
- *BUFFER*: A text work space used to hold and manipulate text records. All editing, except for certain string operations, is done using buffers. Files are read into buffers, which may be mapped to windows for display on the user's terminal while being updated, then are written back out to text files.
- *WINDOW*: A specific region of the user's terminal screen used for text display. Windows may be created, sized, mapped to specific buffers, and deleted.
- *MARKER*: A pointer to a specific character location within a buffer. A marker may be assigned different video attributes for display on the user's terminal, such as blinking, bold, reverse-video, or underline.
- *RANGE*: All characters between two specific markers, including the characters associated with the markers. Ranges may also be assigned different video attributes for display on the user's terminal.
- *PATTERN*: A template for a sequence of characters to be searched for within a buffer. Several VAXTPU built-in procedures and operators are used exclusively when creating patterns. Patterns are handy for doing wild card-type searches. For example, the pattern variable *ur_pat* could be used to search for any string that begins with a less-than sign (<) and ends with a greater-than sign (>):

```
ur_pat := '<' & MATCH( '>' );
```

In this example, the pattern operator ampersand (&) is used with the VAXTPU built-in procedure *MATCH( )* to define the pattern.

Absent from this list of data types are such favorites as floating-point numbers and arrays. (Since VAXTPU is a text-processing language, I presume the developers deemed floating-point unnecessary.) However, programs written in other languages can be accessed through the built-in procedure *CALL_USER*. Floating point can be used, but not without some difficulty. Buffers can be thought of as arrays of text records, although accessing the data via buffers is not as convenient as conventional indexing.

The VAXTPU statements that control the execution of a VAXTPU program are similar to those found in other high-level, structured programming languages. The following are the most commonly used statements.

Procedure statements. The *PROCEDURE*, *ENDPROCEDURE*, *LOCAL*, *ON_ERROR*, and *ENDON_ERROR* reserved words are used to collect a set of VAXTPU statements into a callable routine, define local variables, and define exception processing. The syntax is:

```
PROCEDURE procedure_name
  [(parameter-list)]
    [LOCAL variable_list;]
    [ON_ERROR
       statements;
    ENDON_ERROR;]
    statements;
    ENDPROCEDURE;
```

LOCAL is optional and is VAXTPU's sole variable declaration statement. It defines variable identifiers whose scope extends only to this procedure and to procedures called from this procedure. Global variables are implicitly declared by their use in assignment statements.

ON_ERROR and ENDON_ERROR are also optional and define the processing desired in the event of a VAXTPU-detected error condition. (An example could be a "String Not Found" error be-

ing returned when using the built-in *SEARCH* procedure.)

Interactive access to user-written procedures from EVE is gained with a special command key called the *DO* key. This command makes development and testing easy since the procedures can be compiled and invoked directly from the EVE editor. Even built-in VAXTPU procedures and assignment statements may be executed interpretively in this manner.

Assignment statements. These statements are used to set a value to a variable while implicitly defining the data type of that variable. The syntax is:

```
variable_name := expression;
```

Loop statements. The *LOOP, EXITIF,* and *ENDLOOP* reserved words are used to define a group of VAXTPU statements that will be iteratively executed. The syntax is:

```
LOOP
   [statements;]
   EXITIF conditional_expression;
   [statements;]
ENDLOOP;
```

The statements between *LOOP* and *ENDLOOP* are executed until the result of the conditional expression is true. Note that any statements between *EXITIF* and *ENDLOOP* are not executed after the conditional expression is returned true.

Conditional statements. The *IF, THEN, ELSE,* and *ENDIF* reserved words are used to define groups of VAXTPU statements that will be conditionally executed. The syntax is:

```
IF conditional_expression THEN
   [statements;]
[ELSE
   statements;]
ENDIF;
```

Case statements. The *CASE, FROM, TO, INRANGE, OUTRANGE,* and *ENDCASE* reserved words are used to define groups of VAXTPU statements that will be conditionally executed. The syntax is:

```
CASE variable FROM lower_bound TO
upper_bound
   [case_id_1] : statements;
   [case_id_2] : statements;
    .
    .
   [case_id_n] : statements;
   [INRANGE] : statements;
   [OUTRANGE] : statements;
ENDIF;
```

VAXTPU's *case* construct is similar to the ordinary *case* selector found in other languages. First, in defiance of my earlier syntax examples, the square brackets surrounding the preceding *case* identifiers are a required, not optional, part of the syntax. Multiple *case* identifiers may be listed within each set of brackets, separated by commas.

However, the *case* selectors *INRANGE* and *OUTRANGE* are optional. *INRANGE* will match any *case* within the range of *upper_bound* to *lower_bound* that does not have an explicit *case* identifier listed, and *OUTRANGE* matches anything not within the bounds.

Additionally, the *case* identifiers are restricted to being integer constants or one-character string constants.

Complementing the VAXTPU data types and control statements are 92 built-in procedures (Table 1) that allow file access, manipulation of the various data types, and control of the user's terminal environment. Obviously, I cannot detail all of these procedures here, but I'll describe those I have found most useful in my work.

BEGINNING_OF and *END_OF* each return a marker that corresponds to the first and last positions, respectively, of a buffer or range. They are used in the form:

```
marker := BEGINNING_OF( {buffer |
   range} );
marker := END_OF( {buffer | range} );
```

COPY_TEXT places a copy of the specified string, range, or buffer into the current buffer at the current cursor position. Similarly, the built-in *MOVE_TEXT* copies the specified text, then

VAXTPU built-in procedures

ADJUST_WINDOW
ANCHOR
ANY
APPEND_LINE
ARB
ASCII
ATTACH

BEGINNING_OF

CALL_USER
CHANGE_CASE
COMPILE
COPY_TEXT
CREATE_BUFFER
CREATE_PROCESS
CREATE_RANGE
CREATE_WINDOW
CURRENT_BUFFER
CURRENT_CHARACTER
CURRENT_COLUMN
CURRENT_DIRECTION
CURRENT_LINE
CURRENT_OFFSET
CURRENT_ROW
CURRENT_WINDOW
CURSOR_HORIZONTAL
CURSOR_VERTICAL

DEFINE_KEY
DELETE

EDIT
END_OF
ERASE
ERASE_CHARACTER
ERASE_LINE
EXECUTE
EXIT
EXPAND_NAME

FAO
FILE_PARSE
FILE_SEARCH
FILL

GET_INFO

HELP_TEXT

INDEX
INT

JOURNAL_CLOSE
JOURNAL_OPEN

KEY_NAME

LAST_KEY
LEARN_BEGIN
LEARN_END
LINE_BEGIN
LINE_END
LOOKUP_KEY

MAP
MARK
MATCH
MESSAGE
MOVE_HORIZONTAL
MOVE_TEXT
MOVE_VERTICAL

NOTANY

POSITION

QUIT

READ_CHAR
READ_FILE
READ_KEY
READ_LINE
REFRESH
REMAIN

SAVE
SCAN
SCANL
SCROLL
SEARCH
SELECT
SELECT_RANGE
SEND
SEND_EOF
SET
SHIFT
SHOW
SPAN
SPANL
SPAWN
SPLIT_LINE
STR
SUBSTR

TRANSLATE

UNDEFINE_KEY
UNMAP
UPDATE

WRITE_FILE

Table 1.

DEBUGGING SWAT TEAM

Order Eco-C88 Rel. 4.0 New Modeling Compiler and get C-more at no extra charge!

Seek and Correct

You already know that fast compilation does not mean fast program development. Backing up for bogus error messages and removing the bugs takes time. Eco-C88's "Seek and Correct" three - way error checking finds even the most elusive bugs, clearing the path for swift program development.

Double Barrel Error Checking

Eco-C88 nails **syntax errors** cold and tells you about the error in plain English. And there's no avalanche of false error messages, either. Other compilers can generate up to four times the number of error messages actually present; they leave it up to you to guess which ones are real. You'll be more productive with Eco-C88 because there is no guess work.

Eco-C88 provides ten levels of **semantic error** checking. You can select from almost no checking to the fussiest you've ever seen. Eco-C88's "picky flag" finds subtle errors that slip by other compilers.

Eco-C88 also features:

- All data types, plus ANSI Enhancements
- Robust library, including many new ANSI functions
- CED editor with online function help, split windows, compile-edit-link capability
- New, expanded manual with sample programs for the library functions

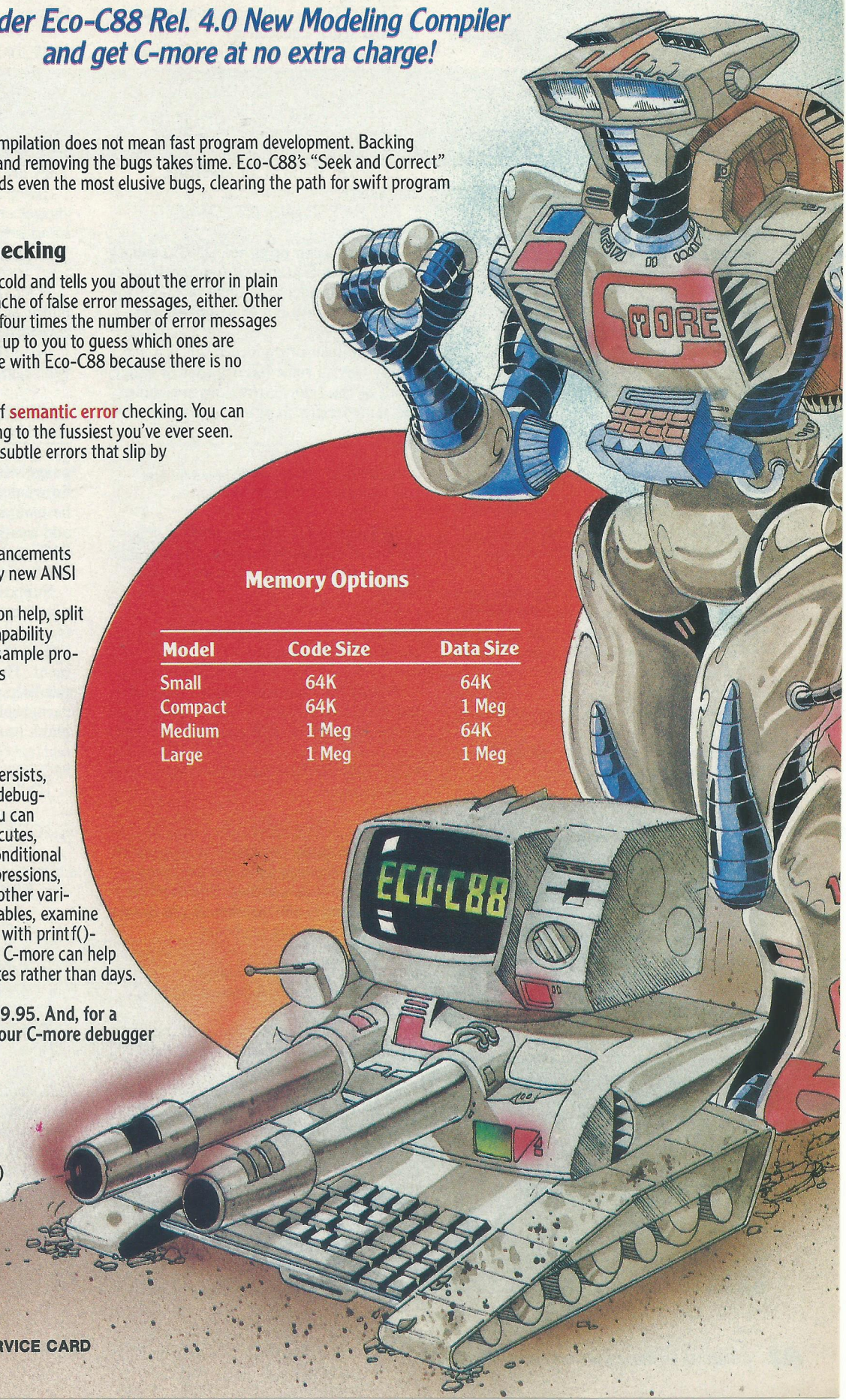
C-more Source Code Debugger

Finally, if a really nasty bug persists, put C-more, our source code debugger, to work. With C-more you can watch your program as it executes, single-step it, set simple or conditional breakpoints, test complex expressions, use variables as indexes into other variables, initialize and trace variables, examine CPU registers, display results with printf()-type options and much more. C-more can help you track down bugs in minutes rather than days.

The price for Eco-C88 is $99.95. And, for a limited time, we'll give you our C-more debugger at no extra charge.

Memory Options

Model	Code Size	Data Size
Small	64K	64K
Compact	64K	1 Meg
Medium	1 Meg	64K
Large	1 Meg	1 Meg



Ecosoft Inc.
6413 N. College Ave.
Indianapolis, IN 46220

(317) 255-6476 (Tech Info)
(800) 952-0472 (Orders)

CIRCLE 31 ON READER SERVICE CARD

deletes the original text. Their formats are:

```
COPY_TEXT( {string | range | buffer } );
MOVE_TEXT( {string | range | buffer } );
```

CREATE_BUFFER is used to define a new buffer for text editing, with the specified name. This buffer may be created empty or you may read into it the contents of a file. *CREATE_BUFFER* optionally returns a value of type *buffer* that may be used later in the program to refer to the newly created buffer. *CREATE_BUFFER* is used in the form:

```
[buffer := ]CREATE_BUFFER( buffer_
   name_string [,filename] );
```

CURRENT_LINE and *ERASE_LINE* are used to refer to the line of text that is currently being edited in the current buffer. *CURRENT_LINE* returns a copy of the current line as a string. *ERASE_LINE* deletes the current line and optionally returns it as a string. Their formats are:

```
string := CURRENT_LINE;
[string := ]ERASE_LINE;
```

EDIT is a powerful procedure that allows you to perform multiple editing functions upon string variables. The string that you wish to edit is passed to *EDIT*, along with one or more keywords that define what actions to take and how to handle quotation marks. *EDIT* is used in the format:

```
EDIT( string, keyword1[,...][keyword2]);
```

where *keyword1* can be any of the following, separated by commas.

Sample USED_BY file

INITIAL_POS_EAST	NAVIGATION
INITIAL_POS_EAST	OUT_DISPLAY
INITIAL_POS_NORTH	NAVIGATION
INITIAL_POS_NORTH	OUT_DISPLAY
INITIAL_REST_COUNT	RESTAURANT_SELECT

Figure 1.

Sample DESCRIPTION file

INITIAL_POS_EAST

Auto's initial east position in degrees.

INITIAL_POS_NORTH

Auto's initial north position in degrees.

INITIAL_REST_COUNT

Initial number of restaurants the vehicle will visit.

Figure 2.

COLLAPSE. Deletes all tab and space characters.

COMPRESS. Replaces multiple or sequences of tab or space characters with a single-space character.

TRIM. Deletes all leading and trailing tab and space characters.

TRIM_LEADING. Deletes all leading tab and space characters.

TRIM_TRAILING. Deletes all trailing tab and space characters.

UPPER. Returns all uppercase characters.

LOWER. Returns all lowercase characters.

INVERT. Toggles the case of all characters.

keyword2 can be either ON (to signify that all quotation marks and apostrophes are to be considered quotes) or OFF (to say that they are not to be considered quotes).

The built-in procedure *FILL* is a handy way to reformat a buffer or range so all of the reformated lines are about the same length. It is used with the format:

```
FILL( {buffer | range}, word_separators,
   l_margin, r_margin );
```

INT and *STR* are the familiar converters of strings to integers and integers back to strings. They are used in the forms:

```
integer := INT( string );
string := STR( integer );
```

POSITION is used to move the editing cursor position to a buffer, marker, range, or window. It is used in the form:

```
POSITION( {buffer | marker | range |
   window } );
```

READ_LINE displays an optional prompt string on the user's output device, reads the response from the keyboard, and returns it as a string. You can optionally specify the number of characters to be read from the keyboard.

```
[response_str := ]READ_LINE
   [( prompt_str [,response_length] ) ];
```

SEARCH returns a range that corresponds to the first occurrence of the specified string or pattern. You must supply *keyword1* to specify whether the search is to be in the *FORWARD* or *REVERSE* direction. You may supply *keyword2* to indicate whether there needs to be an *EXACT* match in terms of character case. *keyword2* may also be set to *NO_EXACT* if an exact match is not needed:

```
range := SEARCH( {string | pattern },
   keyword1 [, keyword2] );
```

SPLIT_LINE breaks the line currently being edited into two lines. This procedure is similar to adding a *Carriage_Return*, *Line_Feed* at the current editing position. It is used in the format:

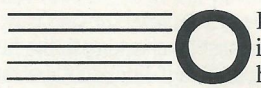
```
SPLIT_LINE;
```

LENGTH and *SUBSTR* are the familiar string operations that return the length of a string and the specified subset of a string or range, respectively. They are used as:

```
integer := LENGTH( string );
string := SUBSTR( {string | range},
   start_position, length );
```

WRITE_FILE writes the specified buffer or range into the specified file. It is used in the format:

```
WRITE_FILE( {buffer | range}, filename );
```



OK, so VAXTPU is a structured, high-level language with a bunch of funny built-ins. So what can you do with it? One of the best ways to begin with VAXTPU is to add a customized editing function to the EVE text-processing interface, which is written in VAXTPU.

Suppose, in the course of updating a document with EVE, you find yourself frequently needing to skip to the first line of the next paragraph in your document. The following VAXTPU procedure can help you do just that (assuming each paragraph is separated from others by blank lines):

```
PROCEDURE eve_next_paragraph
   LOCAL cline;   ! The current line that
      we're editing
   LOOP
      cline := CURRENT_LINE;   ! Get
      the current line
      EDIT( cline, TRIM );   ! Trim off all
      spaces
      MOVE_VERTICAL( 1 );   ! Move
      cursor down to next line
      EXITIF ( cline = "" );   ! Quit if
      previous line was blank
   ENDLOOP;
ENDPROCEDURE;
```

In the *LOOP* in *eve_next_paragraph*, the current line of text in the current active buffer is assigned to the string variable *cline*. All of the spaces are then trimmed from this string, the cursor is moved down to the next line, and *cline* is tested to see if it was a blank line.

The EVE interface can be extended by adding this procedure to its repertoire of

EVEN MORE POWER AND FLEXIBILITY

BRIEF 2.0

Users and industry press alike have unanimously proclaimed BRIEF as the best program editor available today. Now, the best gets better, with the release of BRIEF 2.0.

Straight from the box, BRIEF offers an exceptional range of features. Many users find that BRIEF is the only editor they'll ever need, with features like real, multi-level Undo, flexible windowing and unlimited file size. But BRIEF has tremendous hidden power in its exclusive macro language. With it, you can turn BRIEF into your own custom editor containing the commands and features you desire. It's fast and easy.

Jerry Pournelle, columnist for BYTE magazine summed it all up by saying BRIEF is, "**Recommended. If you need a general purpose PC programming editor, look no further.**" His point of view has been affirmed by rave reviews in C JOURNAL, COMPUTER LANGUAGE, DR. DOBB'S JOURNAL, DATA BASED ADVISOR, INFOWORLD AND PC MAGAZINE.

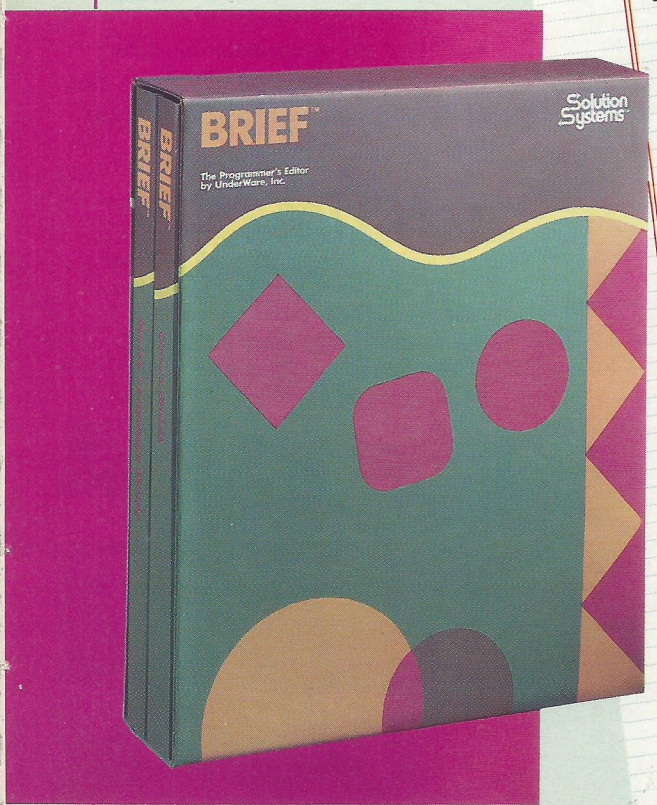
One user stated "BRIEF is one of the few pieces of software that I would dare call a masterpiece." Order BRIEF now and find out why. **BRIEF 2.0 is just $195. If you already own BRIEF, call for upgrade information.**

TO ORDER CALL: 1-800-821-2492 (in MA call 617-337-6963)

As always, BRIEF comes with a 30 day money-back satisfaction guarantee.

Solution Systems™

541 Main St., Suite 410L
South Weymouth, MA 02190
(617) 337-6963



Look at these BRIEF 2.0 enhancements!

Main Features:

- All new documentation with tutorials on basic editing, regular expressions **and** the BRIEF Macro Language.
- Setup program for easy installation and configuration. (Requires no knowledge of the macro language)
- Increased speed for sophisticated operations like Undo and Regular Expression Search.
- Expanded regular expressions, with matching over line boundaries.
- More block types, with marking by character, line or column.
- Command line editing (move cursor, add and delete characters, specify command parameters).
- Support for more programming languages.
- Optional borderless windows.
- Enhanced large display support, including wider displays.
- Reconfigurable indenting for C files (supports most indenting styles).

Plus the basic features that made BRIEF SO popular!

Basic Features:

- Full multi-level Undo
- Windows
- Edit many files at once
- File size limited only by disk space
- Automatic language sensitive indentation

Requires an IBM PC or compatible with at least 192K RAM.

BRIEF is a trademark of UnderWare, Inc.
Solution Systems is a trademark of Solution Systems.

CIRCLE 42 ON READER SERVICE CARD

How A C Programmer Became A Screen Star

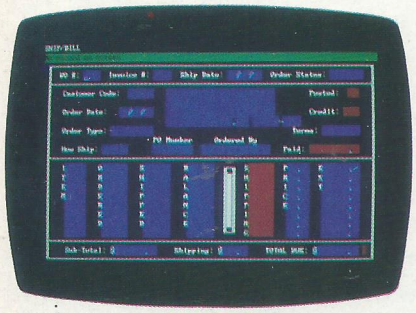
Screens, the Visible Part of Your Program.

A program is often judged by how well the screens are executed. However, the real creativity lies in what goes on behind the screens.

ScreenStar is a product that allows your real creativity to light up the screen. It reduces costly screen, window, and data validation development time.

You Take the Bows, We Write the Code.

Our natural drawing commands allow you to paint any screen imaginable. Press one key when you are satisfied and ScreenStar produces concise, commented, ready-to-compile code. This allows immediate testing of the I/O screens, including smooth, even scrolling between multiple screens.



Create or capture complex screens with data-entry filters built in.

If all ScreenStar did was turn screens into code it would be a useful tool. Yet ScreenStar also permits a wide range of field types. Some of the choices include date, alphanumeric, telephone, yes/no, dollar, time and user-definable fields.

Other valuable data-entry filters are built in, such as required field, display only, and many others. All screen fields are generated with error-checking routines.

ScreenStar Not Only Captures Your Imagination, It Captures Screens.

The memory-resident capture program converts any screen into a ScreenStar file in seconds, including those generated by programs like Dan Bricklin's Demo Program.

ScreenStar Sets the Stage for Windows.

ScreenStar comes with a complete window generating library. You design the help screens and pop-up windows. Essential ScreenStar windowing functions tie them together in one smooth package.

Curtain Call.

They may not ask for your autograph, but they will want to know how you did those screens. Screenstar is more than a screen-painting program. It is a screen processor. No professional programming environment will be complete without this product.

We know you will enjoy using ScreenStar. However, should you give it less than rave reviews, return it within 30 days for a full refund.

★ Interactive screen painting and subsequent code generation.

★ Multiple screen design and scrolling.

★ TSR screen capture program, works with any program including Dan Bricklin's Demo Program.

★ Complete window design including overlapping window functions.

★ Screens are compressed into data structures, and remain a permanent part of the program. No messy data files to look for.

Price - $99

W/Source add $99

Audition Our Product Today. Call:

CIRCLE 43 ON READER SERVICE CARD

ScreenStar is a trademark of Essential Software Inc.
Dan Bricklin's Demo Program is a trademark of Software Garden Inc.

```
!*************************************************************************
PROCEDURE          eve_gen_report
!       This procedure generates the data dictionary report file
!       OUTPUT.MAS from the master input files GENERATED_BY.MAS,
!       USED_BY.MAS, and DESCRIPTION.MAS
!*************************************************************************
  LOCAL cline,              ! current line in buffer gen_buf
                out_buf,                ! output buffer
                var_name;               ! variable name


  tabb := "          ";   ! global variable for blank spaces at
                          ! beginning of each output line

  out_buf := CREATE_BUFFER( "OUTPUT.BUF" );
  gen_buf := CREATE_BUFFER( "GEN_BY_BUFFER", "GENERATED_BY.MAS" );
  use_buf := CREATE_BUFFER( "USE_BY_BUFFER", "USED_BY.MAS" );
  des_buf := CREATE_BUFFER( "DESC_BUFFER", "DESCRIPTION.MAS" );

  LOOP
        POSITION( BEGINNING_OF( gen_buf ) );
        EXITIF ( GET_INFO( CURRENT_BUFFER, "record_count" ) = 0 );
        cline := CURRENT_LINE;
        get_token( cline, 1, var_name );
        POSITION( END_OF( out_buf ) );
        COPY_TEXT( tabb + var_name );
        SPLIT_LINE;
        SPLIT_LINE;
        put_gen_used( var_name );
        put_description( var_name );
  ENDLOOP;

  WRITE_FILE( out_buf, "OUTPUT.MAS" );
  ENDPROCEDURE;
```

Listing 1.

```
!*************************************************************************
PROCEDURE       put_gen_used( var_name )
!       This procedure builds the "Generated by:" and "Used by:"
!       fields in the data dictionary report output buffer.
!*************************************************************************
       LOCAL   gend_by,
                       output_line,
                       used_by;

       gend_by := "";
       used_by := "";

       get_gend_used( gen_buf, var_name, gend_by );
       get_used_used( use_buf, var_name, used_by );
       output_line :=  tabb + "Generated by: " + gen_by;
       pad( output_line, 40 );
       output_line := output_line + "Used by: " + used_by;
       COPY_TEXT( output_line );
       SPLIT_LINE;

       LOOP
              get_gend_used( gen_buf, var_name, gend_by );
              get_used_used( use_buf, var_name, used_by );
              EXITIF (gend_by = "") AND (used_by = "");
              output_line :=  tabb;
              pad( output_line, 23 );
              output_line :=  output_line + gen_by;
              pad( output_line, 48 );
              output_line := output_line + used_by;
              COPY_TEXT( output_line );
              SPLIT_LINE;
       ENDLOOP;

       SPLIT_LINE;
       ENDPROCEDURE;
```

Listing 2.

DAN BRICKLIN'S DEMO PROGRAM ONLY $74.95

Read what they're saying about this popular program for prototyping and demo-making:

"A winner right out of the starting gate. After you use DEMO once, you'll wonder how you got along without it."
—PC Magazine

"Everybody who writes software, either commercially or for in-house applications, should immediately order a copy. Period. No exceptions."
—Soft·letter

Product of the Month
—PC Tech Journal

Thousands of developers and most of the largest and best known software companies are using this program. You can, too. Act now!

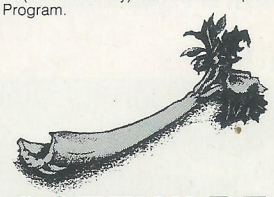
NEW TUTORIAL! JUST $49.95

The perfect companion to the Demo Program. The Tutorial helps you learn the ins and outs of its basic and advanced features. Complete with a 96 page manual containing step-by-step instructions, diskette, and function key template.

ORDER NOW!
1-800-CALL-800 x8088

VISA MasterCard AMERICAN EXPRESS

Use 800-number for orders only.
Questions, special shipping, etc., call **617-332-2240.**
No Purchase Orders. Massachusetts residents add 5% sales tax. Outside of the U.S.A., add $15.00.
Requires 256K IBM PC/Compatible, DOS 2.0 or later. Supports Monochrome, Color Graphics, and EGA Adapters (text mode only). The Tutorial requires the Demo Program.



SOFTWARE GARDEN, INC.
Dept. L
P.O. Box 373, Newton Highlands, MA 02161

CIRCLE 44 ON READER SERVICE CARD

CQL QUERY SYSTEM

Portable Application Support System

Both for $395.00

Add SQL compatible ad-hoc query capability to your new and existing applications

Includes CQL Interpreter, embedded CQL Library, Portable Windowing System, Screen I/O, Report, and Form Generation Systems.

Complete C Source Code Included.

Hardware Independent
Interfaces provided for IBM/screen memory, IBM/BIOS, MS-DOS generic (ANSI.SYS), and Xenix (table driven multi-terminal interface adaptable to other large systems).

Compiler Independent
Tested with Microsoft V4.0, Lattice V3.1, Lattice V2.15 (reduced system), Aztek, and Xenix System V Version 1.2.

File System Independent
Interfaces provided for C-tree (trademark of Faircom) and BTRIEVE (trademark of SoftCraft Inc.).

Rich Set of Data Types
Data types include 8, 16, and 32 bit binary, 32 bit Monetary (integer), 32 and 64 bit floating point, 32-bit date, 32-bit time.

CQL Features
Automatic optimization and index use
Link detection and alias mechanism
Table driven parser is easily modified
'unique' and 'order by' clauses
'in', 'any', and 'all' operators
Correlated queries & interblock references
'count', 'sum', 'min', 'max', 'avg' functions
Generates indices for interblock links.

Includes:
90 day support and free updates on disk
Free updates for any bug fixes, forever

Machine Independent Software Corporation
1415 Northgate Square #21B
Reston, Virginia 22090
(703) 435-0413
9AM-1PM, 5PM-8PM EST

CIRCLE 45 ON READER SERVICE CARD

editing commands. You could even map this procedure to its own function key to make it easier to invoke.

A more substantial illustration of VAXTPU programming can be found in the following case history. The company I work for writes a large amount of software for the U.S. government. We recently found ourselves with only two weeks to input, format, and publish a 500-page data dictionary. In this document we had to list each variable used in our application program and provide a textual description and list of the software functions that generated and used each variable.

Although the task was not particularly difficult, the extremely short deadline presented a real challenge. The large amount of time needed to set up a conventional data base, teach people to input the data, and generate a suitable report prevented us from going that route.

Instead, we instructed everyone to use his or her favorite text editor to start inputting data into three text files: GENERATED_BY, USED_BY, and DESCRIPTION. These files were produced with very little training. The data was separated from the variable name by several spaces (or a blank line, in the case of the description). Figures 1 and 2

```
!*****************************************************************************
PROCEDURE    get_gend_used( buffername, var_name, return_string )
!      This procedure returns the next "Generated by" or "Used by"
!      field associated with var_name from buffer buffername.
!*****************************************************************************
        LOCAL   cline,
                search_range;  ! range returned from SEARCH built-in

        ON_ERROR
                IF ( error = tpu$strnotfound ) THEN
                        search_range := 0;
                ENDIF;
        ENDON_ERROR;

        return_string := "";
        POSITION( BEGINNING_OF( buffername ) );
        search_range := SEARCH( var_name, FORWARD );
        IF (search_range <> 0 ) THEN
                POSITION( search_range );
                cline := ERASE_LINE;
                get_token( cline, 2, return_string );
        ENDIF;
        ENDPROCEDURE;
```

Listing 3.

```
!*****************************************************************************
PROCEDURE    get_token( in_string, n , token_string )
!       This procedure returns the nth token from the
!       input string
!*****************************************************************************
   LOCAL counter,      ! These are local copies of the input
                 temp_str,       ! parameters that we can alter
                 space_index,  ! position of 1st space in temp_str
                 str_len;        ! length of string to extract

   temp_str := in_string;
   counter := n;
   LOOP
           EDIT( temp_str, TRIM );
           space_index := INDEX( temp_str, " " );
           EXITIF ( counter = 1 );
           str_len := LENGTH( temp_str ) - space_index;
           temp_str := SUBSTR( temp_str, space_index, str_len );
           counter := counter - 1;
   ENDLOOP;

   IF ( space_index = 0 ) THEN
           token_string := temp_str;
   ELSE
           token_string := SUBSTR( temp_str, 1, space_index - 1 );
   ENDIF;

   ENDPROCEDURE;
```

Listing 4.

show excerpts from sample USED-BY and DESCRIPTION files.

Standard VAX utilities were then used to merge the individual lists into three master lists and sort the GENERATED_BY and USED_BY lists. The final report was generated using a set of VAXTPU procedures similar to those shown in Listings 1-5.

The top-level procedure (*eve_gen_report*) starts by creating an empty buffer in which to place the report (*out_buf*), then continues by reading the three master lists into the three buffers called *gen_buf, use_buf*, and *des_buf*. It then enters its main loop and iteratively performs four actions:

1. Checks to see if it is done by testing for any more records in *gen_buf*.
2. If not, it retrieves the next variable name from *gen_buf*.
3. Copies the variable name into the output buffer.
4. Calls two procedures, *put_gen_used* and *put_description*, to build the other fields in the output buffer.

When *eve_gen_report* is done, it merely needs to write the output buffer out to the file OUTPUT.MAS with a call to the VAXTPU built-in WRITE_FILE.

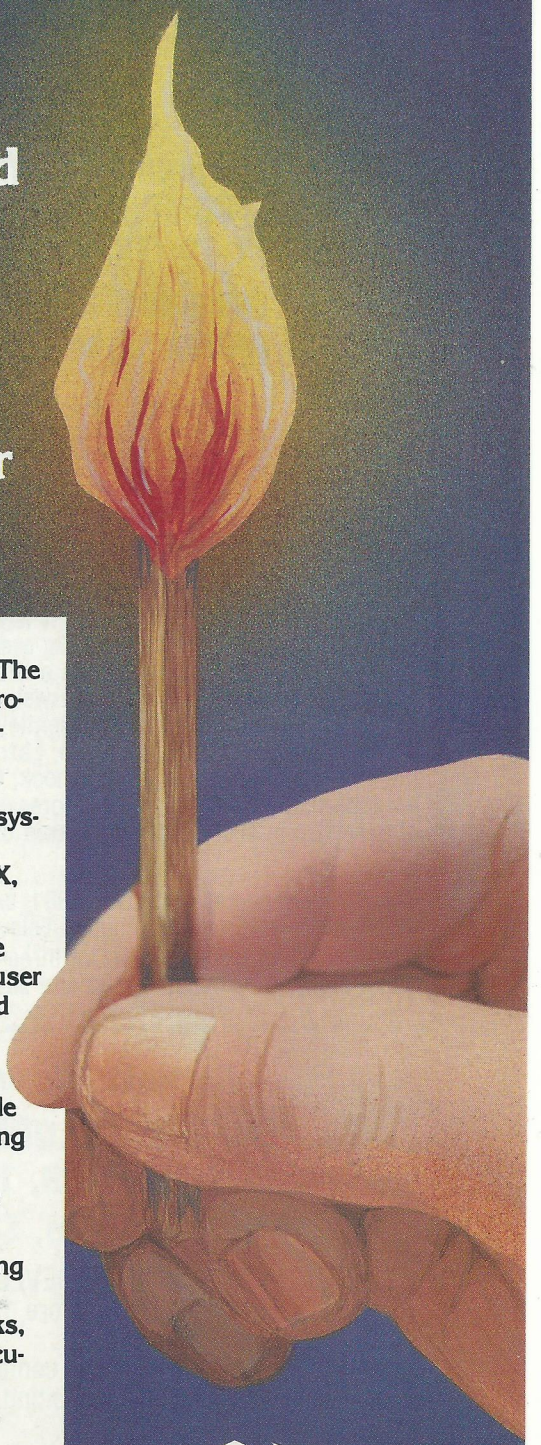
As we've seen, VAXTPU provides an interesting alternative for processing textual data. File I/O is easy and VAXTPU's extensive built-in procedures provide a powerful set of tools for manipulating text. Programming in VAXTPU is fast, and the interactive debugging provided by the EVE interface makes the development almost painless.

I have used VAXTPU programs to format technical publications, generate column-oriented reports from line-oriented input data, recursively analyze compiler listings to produce hierarchical calling trees, and even estimate module memory- and stack-size requirements.

You must remember, though, that VAXTPU is designed as a text-processing program. You can easily be fooled by its power and try to use it in inappropri-

Unmatched.

If you want unmatched performance and portability, we have it. The hottest file handler and report generator on the market.



The c-tree file handler offers unmatched file accessing speed. The r-tree report generator makes producing reports a snap. Both packages offer unmatched portability. Thousands of programmers are using these packages in over 50 system environments: DOS, UNIX, XENIX, OS/2, MACINTOSH, VAX, TOWER and ..... YOURS.

More for your money • complete C-source code • single and multi-user capability • no royalties • unlimited free technical support • port to all machines.....for one price.

c-tree features • fixed and variable length data records • record locking • variable length keys and key compression • overcomes OS file limit ..... and more.

r-tree features • no printer spacing charts • change reports without recoding • unlimited control breaks, accumulators and virtual field calculations • powerful search, select and sort capabilities over multiple files.....saves days of coding.

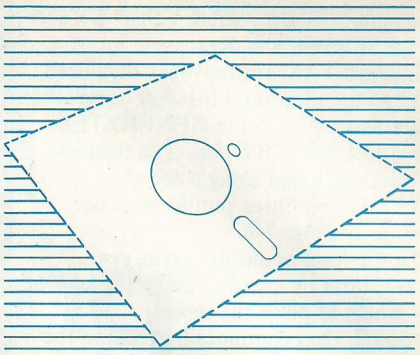
FairCom's unmatched products will work for you. Order c-tree today for $395, r-tree for $295. When ordered together, r-tree is only $255. For VISA, MasterCard and C.O.D. orders call (314) 445-6833. For c-tree benchmark comparisons, write us at 4006 West Broadway, Columbia, MO 65203.

CIRCLE 46 ON READER SERVICE CARD

c-tree™/r-tree™
By FairCom
4006 W. Broadway Columbia, MO 65203

UNIX is trademark of AT&T, MACINTOSH is trademark licensed to Apple Computer Company,
VAX is trademark of DEC, TOWER is trademark of NCR, XENIX is trademark of Microsoft

Now!



SOFTWARE DEVELOPMENT '88

A Conference Brought To You By The Editors of AI EXPERT, COMPUTER LANGUAGE, and UNIX REVIEW!
February 17-19, 1988 San Francisco, Calif.
The most comprehensive conference ever held on *practical aspects* of software development

Never before have so many leaders in the software development field gathered for one event. SOFTWARE DEVELOPMENT '88 is an exciting conference and trade show on the latest innovations and developments in programming. The conference is designed to teach you practical, advanced techniques you can use in your work each day.

SOFTWARE DEVELOPMENT '88 Faculty is a virtual Who's Who of the programming community
Nearly 100 speakers will address a broad range of software development topics. Our well-known and respected faculty includes: Jon Bentley, Dick Gabriel, Bill Gates, P.J. Plauger, Esther Dyson, Larry Constantine, Gary Kildall, Philippe Kahn, John Warnock, Charles Duff, and Kouichi Kishida. The speakers represent a wide spectrum of expertise in every significant area of programming.

Register Now!
Call Bill Rutledge at (415) 397-1881, ext. 393.
Or complete and mail the coupon below.
The cost for this comprehensive three-day event is only $695. Sign up by December 31 and save $120!

Focus on Languages
At Software Development '88, you will find six complete tracks of courses to choose from, all in areas of interest to developers: Languages, Artificial Intelligence, Design Methodologies, C, Graphics, Programming Tools and Issues, and Data Base Design. The following list of lectures and workshops in the Languages track is just an example of the kind of topics and speakers you will find in every track:

	Lectures
■ Alan Kay	■ User Interfaces and How People Learn
■ Philippe Kahn	■ The Future of Language Development
■ Charles Moore	■ The Four Forths of Charles Moore
■ Bill Gates	■ New Language Environments: A Vision
	Workshops
■ David Boswell	■ A Quick Primer on Compiler Construction
■ Rob Dickerson	■ Assembly Language Programming on the PC
■ Avram Tetewsky	■ Ada for Conventional Language Programmers
	■ Introduction to Ada Tasking
■ John Malpas	■ PROLOG as an Application Language
■ Don Baccus	■ Modula-2 for Systems Programming
	■ You Can't Do That in Pascal
■ William Barrett	■ Special-Purpose Languages: Applications of a Compiler Generator
■ Charles Duff	■ Software Development with Actor
■ Don Nelson	■ Programming in COBOL-85

Full tracks on six other areas of interest to developers will also be available to choose from: Languages, Design Methodologies, Programming Tools and Issues, Data Base Design, C Language, and Graphics.

Your registration entitles you to attend six one-hour lectures choosing from 30 courses conducted by industry leaders. In addition, you can select six longer, in-depth workshops from a list of nearly 60 different topics. The workshops will focus on practical techniques and the latest developments you can apply to your work right away. You won't be just an observer. Lively debates will take place in panel discussions and problem-solving workshops.

Register Now!
SOFTWARE DEVELOPMENT '88
February 17-19, 1988
San Francisco, Calif.

☐ Register me for SOFTWARE DEVELOPMENT '88.
_Payment enclosed ($575 before 12/31)
_Send invoice
☐ Send me more details on the conference.
☐ Send me information on exhibiting at the trade show.

Name _____
Company _____
Address _____
City, State, Zip _____
Phone ( ) _____

Mail to:
Seminar Department
Miller Freeman Publications
500 Howard Street
San Francisco, CA 94105

CIRCLE 28 ON READER SERVICE CARD

ate applications. I would love for someone to take the basic concepts provided by VAXTPU and extend them to include support for other high-level data structures (such as floating-point numbers and arrays). To me, that would be close to an ideal programming environment.

Leonard Struttmann has a B.S. degree in electrical engineering from the University of Missouri at Rolla, Mo. He has written real-time avionics software for more than eight years for McDonnell-Douglas, Sperry Corp., and Rockwell. His work includes providing support and productivity tools for use in software development VAX minicomputers.

```
!*****************************************************************************
PROCEDURE         put_description( var_name );
!    This procedure builds the "Description"  fields in the
!    data dictionary report output buffer.
!*****************************************************************************
  LOCAL cline,
                  des_range,        ! description to be
                                    !    transferred
                  m1,               ! marker at beginning of
                                    !    description
                  m2,               ! marker at beginning of
                                    !    description
                  search_range;     ! range returned from SEARCH
                                    !    built-in

  ON_ERROR
          IF ( error = tpu$strnotfound ) THEN
                  search_range := Ø;
          ENDIF;
  ENDON_ERROR;

  POSITION( BEGINNING_OF( des_buf ) );
  search_range := SEARCH( var_name, FORWARD );
  IF (search_range <> Ø ) THEN
          POSITION( search_range );
          cline := ERASE_LINE;      ! delete the variable name
          cline := ERASE_LINE;      ! delete the blank line
          m1 := MARK( NONE );       ! mark beginning of description

          LOOP                      ! find next blank line
                  MOVE_VERTICAL( 1 );
                  cline := CURRENT_LINE;
                  EDIT( cline, TRIM );
                  EXITIF ( cline = "" );
          ENDLOOP;

          m2 := MARK( NONE );       ! mark end of description
          des_range := CREATE_RANGE( m1, m2, NONE );
          POSITION( END_OF( out_buf ) );
          MOVE_TEXT( des_range );   ! move the description to
                                    !    out_buffer
  ENDIF;
  ENDPROCEDURE;
```

Listing 5.

MetaWINDOW

Power Graphics for your PC!

PC TECH JOURNAL

"Product of the Month"
"... a technological tour de force for fast PC graphics."

NO ROYALTIES!

MetaWINDOW is an advanced, high performance graphics development toolkit which bridges the gap between low-level graphic primitive libraries and pre-packaged window managers.

Unparalleled Performance!

MetaWINDOW provides an expanded set of graphic drawing functions, plus the added functionality and performance required for designing multi-window desktop applications.

- auto-cursor tracking
- pull-down menus
- pop-up windows
- comprehensive graphic functions
- multiple fonts

Enhanced Features!

- Display multiple bitmap or "filled-outline" fonts.
- Face fonts for bold, italic, underline or strike-out stylings.
- Full "RasterOp" transfer functions for writing, erasing, rubberbanding or dragging: lines, text, icons, bit images and complex objects.
- Create pop-up menus, windows and icons.
- Supports IBM's new PS/2 VGA and MCGA graphics.

MetaWINDOW comes complete with langauge bindings for 16 popular C, Pascal and Fortran compilers, plus dynamic runtime support for over 40 graphics adaptors and input devices.

MetaWINDOW
Advanced Graphics Toolkit
*4 disks, 3 260 page manuals - $195**

NEW! - TurboWINDOW/C
*All the features of MetaWINDOW for Borland Turbo C! - $ 95**
** Plus $5.00 shipping and handling*

TO ORDER CALL 1-800-332-1550
For information or in CA call 408-438-1550

METAGRAPHICS
SOFTWARE CORPORATION
269 Mount Hermon Road
Scotts Valley, CA 95066

CIRCLE 48 ON READER SERVICE CARD

PRODUCT BINGO

By Johanna Kleppe

Each month Product Bingo features the latest in new software and hardware products of interest to COMPUTER LANGUAGE *readers. Send new product information to Regina Starr Ridley, Editor,* COMPUTER LANGUAGE, *500 Howard St., San Francisco, Calif. 94105.*

C Dick, C Jane, C Spot...

C Spot Run is a set of routines with text windowing, simple graphics, and string manipulation. $15; $50 (with source code); $75 (source code and license for commercial use).

New Dimension Software, 23 Pawtucket Dr., Cherry Hill, N.J. 08003, (609) 424-2595.

CIRCLE 130 ON READER SERVICE CARD

SolveIT!

SolveIT is a numerical analysis tool with a mouse interface and an overlay structure for memory conservation, all supplemented by a technical reference manual. $129

Structured Scientific Software, 1509 Queen Ave. SW, Albany, Ore. 97321, (503) 926-8599.

CIRCLE 131 ON READER SERVICE CARD

Microsoft maintains

Microsoft's FORTRAN Optimizing Compiler v. 4.01 has faster *LOG* and *EXP* functions. Upgrade to v. 4.01 free to v. 4.0 owners, $150 to owners of earlier versions.

Microsoft Corp., 16011 NE 36th Way, Box 97017, Redmond, Wash. 98073-9717, (206) 882-8080.

CIRCLE 132 ON READER SERVICE CARD

Compiling to your advantage

AdaVantage, an Ada development system, compiles over 1,000 lines per minute and implements Ada features like tasking, exceptions, and separate compilation.

Meridian Software Systems, 23141 Verdugo Dr., Ste. 105, Laguna Hills, Calif. 92653, (714) 380-9800.

CIRCLE 133 ON READER SERVICE CARD

Public display of a function

C-Display Manager v. 3.2 has a screen generator and demo utility (*DisplayExpress*), and a library of display and graphics management functions (*C-Display Librarian*).

Sydetech, 43-23 Colden St., #17C, Flushing, N.Y. 11355, (718) 886-0221.

CIRCLE 134 ON READER SERVICE CARD

Motherboard knows best

ZEOS 386/M is a 80386 based motherboard that supports existing standard peripheral cards and provides up to 16 MB of memory on a 32-bit memory bus.

Zeos International, 530 5th Ave. NW, St. Paul, Minn. 55112, (612) 633-4591.

CIRCLE 135 ON READER SERVICE CARD

What's up, AVDOC?

AVDOC is an on-screen reference guide for C programmers with a pop-up calculator that figures decimal, hexadecimal, octal, and ASCII values. $49

Avocet Systems, 120 Union St., P.O. Box 490, Rockport, Maine 04856, (800)448-8500.

CIRCLE 136 ON READER SERVICE CARD

Cross-compiler for Z8

MICRO/C-Z8, a support tool for Zilog Z8, has C access to all on-chip peripherals by name and run-time exception checking of math and memory access operations. $750

Micro Computer Control, P.O. Box 275, Hopewell, N.J. 08525, (609) 466-1751.

CIRCLE 137 ON READER SERVICE CARD

Ecosoft on the rise

Ecosoft's v. 4.0 of the *Eco-C88* C compiler uses syntax parsing techniques and provides a user-selectable lint facility. *C-More* is a source code debugger used by menu or Quickkeys selection. $99.95 (Eco-C88); $49.95 (C-More)

Ecosoft Inc., 6413 N. College Ave., Indianoplis, Ind. 46220, (316) 255-6476.

CIRCLE 138 ON READER SERVICE CARD

Get out your best SilverWare

SilverComm, a communications library for Clipper, dBASE III Plus, FoxBASE, dBXL, and QUICKSILVER, has a library and high-level communications i/o functions. $149.95

SilverWare Inc., P.O. Box 781143, Dallas, Tex. 75378, (214) 902-0227.

CIRCLE 139 ON READER SERVICE CARD

Form and content

cForm is an application development environment with visual forms-oriented application tools. Licenses: $295 (power user), $798 (developer), $15—$150 (run time).

Comsure Computer Services Inc., P.O. Box 235, Chatham, N.J. 07928, (201) 273-8973.

CIRCLE 140 ON READER SERVICE CARD

Toolboxes are opened

Turbo BASIC Toolboxes are for data base, editor, and telecommunications development. $99.95

Borland International, 4585 Scotts Valley Dr., Scotts Valley, Calif. 95066, (408) 438-8696.

CIRCLE 141 ON READER SERVICE CARD

Breakthrough in interface management. Generate C code from Dan Bricklin's Demo screens. Date fields. Full color support. Money fields. Fully programmable field behavior. Scrolling text within fields. Calculator style numeric input. User definable entry validation. Field marking. Orthogonal field movement. Specify fields by number or location. Source code included. Screen sizes limited only by memory. Interfaces with db_VISTA and other libraries. Text style numeric input. Input masking. List fields. Create spreadsheets. Includes Look & Feel screen designer. Integer fields. String formatting commands. Date and time validation functions. Generate C code with Look & Feel screen designer. Supports automatic vertical and horizontal scrolling. Clean screen machine. Number of fields per screen limited only by memory. Fast screen development. String fields. Easy to modify. Fast screen painting. Bind as much data as de sired to fields. Numeric data entry with commas. Ask a bout our linear pro gramming library. Hexadecimal fields. Long fields. Yes or No fields. Float fields. Quick C. Read only fields. Speaker functions. Lattice. Create reports. Slug. Numeric validation routines. Codename Validate data at the keystroke level. Customize screens and menus at run time. 30 day money back guarantee. Gen eric data pointer. Rich assortment of editing commands. Easy to learn. Pop-up windows. Assign validation data to fields. Corporate C credentials. Pull down menus. Sup ports EGA 43 line mode. All functions are kept in separate modules. Full C style function reference. Pop-up prompt and message functions. Numeric range checking. No royalties. Descrip tive function names. Date and time conversion routines. Capture screens from existing programs. Nest screens as deep as desired. Easy to main tain. Run time error checking. Date and time conver sion functions. Screen definition language based on C's printf. Time fields. Ful ly definable borders. The current field can be automati cally highlighted. Create reports. Exploding borders. Convert old programs to C. Borders with titles. Color map enables use of logical colors. Toll-free telephone support line. 24 hour bulletin board. Automatically detects type of monitor being used. ANSI driver included. Screen and field definitions. Uses device drivers for portability. View text in pop-up windows. Read only fields. Rich assortment of editing commands. Pass word entry fields. Includes ROM BIOS driver. Fields can support any data type. Scrolling/ paging functions included. Specify writeable and non-writeable positions within fields. Customizable borders. Se lect different cur sor types. Supports CGA, EGA, and monochrome. Aztec. In cludes functions for writing to the display. In cludes an ANSI device dri ver. A vari ety of keyboard functions. Lined borders. Multi-level menuing systems. Borders with scroll lights. Vid eo RAM dri ver included. New device drivers can be cre ated. Color map enables log ical use of colors. Borders with prompt lines. Fully inte grated help system. Create as many screens as needed. Create data entry screens. Easy to follow man ual. Professional customer sup port. Includes higher level functions. Device drivers swappable at run-time. Context sensitive help system. Cross referenced help screens. Protected fields. Object-oriented design. Read in screen defini tions from disk files. Digitally mastered. Assign prompt strings to fields. UNIX. No run-time license. Numeric range checking. Unified field theory. Full printf % substitution within screen definitions. Supports all memory models. C Bricklin run. Turbo C. 24 hour bulletin board. Higher level functions included. Object-oriented design. All library functions are kept in separate modules. Nest screens as deep as desired. Design screens with Look & Feel screen designer. New device drivers can be created. Create as many screens as needed. No run-time license. Hexadecimal fields. Prefer red by professionals and consultants. Microsoft. Cross referenced help system. by space aliens. Generate C code with Look & Feel screen designer. Context sensitive help system. Scroll lights. Read in screen definitions from disk files. Automatic vertical and horizontal scrolling. Batteries not included. Double and single line borders. Cross-referenced help system. Save and restore regions of the display. Nested menus. Quick C. Create screens from ASCII files. Easy to learn and use. Horizontal and vertical scrolling. Used by consultants and corporations worldwide. Easy to maintain. Professional documentation. Screen designer creates readable C code. Portable. Easily modifiable functions. No royalties. Source code included. Turn Dan Bricklin slides into C. Professional support. Interface examples for data base management. Validation at keystroke level. Vast integrated and indexed context-sensitive help system. Save and restore regions of the display. Now supporting Quick C, Turbo C, Aztec, Lattice, Microsoft, UNIX and others. And that's not all. Call for demo.

C-scape 2.0

with

NEW!

Look & Feel™

The state-of-the-art interface management system preferred by professional C programmers and consultants worldwide.

Look & Feel

- WYSIWYG screen design tool
- Generates readable C code
- Create menus and data entry screens
- Define fields of any type
- Variables, prompts, and validation
- Line draw and erase
- Block, move, cut, paste, copy
- Horizontal and vertical scrolling
- Edit Dan Bricklin Demo slides
- Full color support
- Fast, easy, and fun to use
- Includes help
- Full-feature demo available

C-scape 2.0

- Windows, windows, windows
- Menus, menus, menus
- Vast help system
- Create any type of field
- Data entry and validation
- Smart borders
- Extensive function library
- Swappable device drivers
- Easy to learn and use
- Easy to maintain and modify
- Unsurpassed flexibility
- Professional manual
- No royalties; no run-time license
- Source code included
- Demo package available

Oakland Group, Inc.

675 Massachusetts Avenue
Cambridge, MA 02139-3309

800-233-3733 CALL
617-491-7311 NOW

MasterCard VISA

PC/MS-DOS $279, plus shipping (includes C-scape, Look & Feel, source, manual and support). UNIX/others call. 30-day review.

CIRCLE 49 ON READER SERVICE CARD

Ask for a FREE REPORT: "Interviews with Authors & Users:
How Can These Products Raise Productivity and Help You Write Better Programs?" from
THE PROGRAMMER'S SHOP

A RECONFIGUREABLE PROGRAMMER'S EDITOR WITH FULL SOURCE CODE

ME is a high quality programmer's text editor written specifically for the IBM PC. It contains features only found in the more expensive programmer's text editors. These features include:

- **Multiple Resizeable Windows**
- **Column cut and paste**
- **Line marking for source code**
- **Regular Expressions**
- **C-like Macro Language**
- **Reconfigurable Keyboard**
- **Capture your DOS session**
- **EGA Support**
- **Run your compiler and examine errors**
- **CTAGS**

This is the ONLY editor which has the features of the higher priced editors AND comes with complete source code.

New commands and features may be added to the editor by writing programs in its macro language. The language resembles the C programming language, and comes with a rich set of primitives for handling strings and changing the editor environment, plus most of the flow-of-control constructs that come in C *(for, while, if, break, continue, switch)*.

The code is written in standard C, with several key library routines written in assembler for speed. The source code option is perfect for OEMs and VARs who want to add editing or word processing capabilities to their applications. Using the macro language, people have written packages for DBASE, computer aided software engineering, and graphics.

With your purchase, we include macros for incremental search, C templates, abbreviation facility, and others. User-written macros are constantly being added to our library.

Here is a piece of a macro to perform incremental searching :

```
init()
{
 assign_key("i_search", CTRL_S);  /* bound to <CTRL> S */
}
i_search()
{
 int c, len;
 string pattern;
 save_position();
 while (c != '\E') {
   message(strcat("Incr Search -- ", pattern));
   if ((c = get_tty_char()) == '\b') {   /* backspace */
     if ((len = strlen(pattern)) > 1)
       pattern = substr(pattern, 1, len - 1);
     else
       pattern = "";
     restore_position();          /* go to where we started from */
   }
   else if (c == 7) {             /* <CTRL> G means abort */
     for (len = strlen(pattern); len > 0; len = len - 1)
       restore_position();        /* restore to start position */
     break;
   }
 .......
```

Price for editor and on-line documentation ---- $39.95
Price for editor with complete source ---------- $94.95

MAGMA SYSTEMS

138 - 23 Hoover Avenue Jamaica, New York 11435
For more information, call: (201) 792 - 3954

P.S. - ME is not the same editor as MicroEmacs

ACCELERATE YOUR C DEVELOPMENT
*TRANSLATE** FORTRAN TO C

- **Maximize the vast resources of FORTRAN** while moving up to C. Speed up new C development and avoid reinventing the wheel.
- Use ***RTC Plus*** to translate FORTRAN code and libraries — and **maintain code with greater ease and flexibility** in C.
- **Source code to C libraries is included.**
- ***RTC Plus* supports standard FORTRAN-77** as well as some DEC VAX extensions (excluding FORTRAN I/O, character and complex statements/expressions). Over 95% of STUG's RATFOR is also supported. ***RTC Plus*** generates K&R C.
- **Finally a cost-effective method of conversion into C.**

ORDER TODAY!

RTC Plus
DEMO $10
MS-DOS $450
PS PRICE $399

Translate: "To convey to heaven without natural death."

COBALT BLUE
1683 MILROY, SUITE 101, SAN JOSE, CA 95124
408-723-0474

Now you see it; Now you don't!

TSR's made easy with /*resident_C*/

Finally! You write your C program and we make it resident. No strings attached. Want to process interrupts also? No problem. **/*resident_C*/** is a set of C functions that enable you to process interrupts and/or make your programs "terminate stay resident" like Sidekick.

We've done the research, testing, and grunt work to make your TSR programs safe, compatible and easy. Be aware of the "ifs, ands and buts" when writing interrupt handling software. Only **/*resident_C*/** has all that you need to write reliable interrupt handlers without the worries.

We don't ask you to trust us either. Our manual explains what we are doing very clearly and our documented source code is available. In addition, our working demos give you clear examples of resident programs, interrupt handlers and resident libraries. There is no other product that can do what **/*resident_C*/** can do for you.

/* resident_C */ is the perfect complement to the other library products available from Essential Software.

ESI

List: $99 / with Source add $99
PS: $79 / with Source add $79

Call or Circle Reader Service Number for Your FREE Study.

HOURS: 8:30 A.M.-8:00 P.M. E.S.T.
5-L Pond Park Road, Hingham, MA 02043
Mass: 800-442-8070 or 617-740-2510 9/87

MasterCard **800-421-8006** VISA

THE PROGRAMMER'S SHOP ™
Your complete source for software services and answers

CIRCLE 50 ON READER SERVICE CARD

C-macs

System requirements: UNIX, Ultrix, Microsoft Corp.'s XENIX
Price: $645 for IBM PC/XT/RT or compatibles and Sun 2 and 3 workstations
Available from: UniPress Software Inc., 2025 Lincoln Hwy., Edison, N.J. 08817, (201) 985-8000
Support: Telephone support available

Emacs, like UNIX, Forth, LISP, and a few others, is one of those extensible, interactive software systems that tends to incite strong loyalty in programmers. Basically, it is a screen-oriented text editor that gives the user power to define complex editing operations and set up mappings from keystrokes to operations. With Emacs, the user can associate nearly any sequence of keystrokes with nearly any editing operation. In fact, Emacs is as much a language for designing editors as it is an editor itself.

Emacs traditionally has had strong ties to the LISP culture, and the language used to build complex editing commands within Emacs is very much like LISP. Since Richard Stallman invented Emacs at Massachusetts Institute of Technology, Cambridge, Mass., in the early 1970s, several recreations have appeared. John Gilmore's freeware GNUemacs is popular, as is another free version called microEMACS, written by David Conroy.

The UniPress Emacs discussed in this review is based on a version written by James Gosling when he was at Carnegie-Mellon University, Pittsburgh, Pa., to run under the UNIX operating system. UniPress has developed versions to run under either UNIX or MS-DOS.

Before you begin to think the title of this article contains a typographical error, I should explain that C-macs is the name UniPress gives to a package of C-oriented extensions to Emacs available since early 1987. In this review, Emacs will mean the underlying editor and C-macs the extension package, although sometimes the distinction is fuzzy. Besides strictly focusing on the C-macs package, this review also includes some evaluation of UniPress Emacs and compares Emacs and more traditional screen-oriented editors such as vi.

For a single-user computer such as an IBM-PC/AT or a Sun 3 workstation, a binary license for C-macs costs $645, plus 25% for support (including a year's free updates). The underlying Emacs is included in the price, and if you already have it a credit is available. Although UniPress Emacs is available for MS-DOS, you must have some version of UNIX (such as XENIX) to run C-macs. The version I tested was the one for the Sun 3 workstation.

What kind of C-oriented extensions does C-macs provide? According to the blurb in the UniPress Emacs newsletter, C-macs includes:

- A menu user interface
- Automatic expansion of C control structures
- An electric C mode which provides automatic indentation, parenthesis matching, etc.
- A tags facility similar to that in vi for keeping track of which functions are in which files
- A keyword description data base
- Compile-time aids such as an error message parser.

Furthermore, the user's manual states, "you do not need to be familiar with Emacs to Use this C-macs." This sounds promising. I do a lot of C programming using vi as the editor and have often wished for a little more C-specific intelligence in the editor. The question is, how does this work in practice?

First of all, the documentation is excellent. The C-macs user's manual can be read as a tutorial and the Emacs reference guide has over 700 pages of Emacs function descriptions and variables (in alphabetical order) in a format similar to UNIX man pages. The C-macs user's manual is self-contained and describes the Emacs commands needed for basic editing operations such as creating and deleting buffers and windows, moving the cursor around, and searching, deleting and inserting text, as well as the C-specific C-macs features.

Emacs is fundamentally different from vi in that vi distinguishes between a command mode and an insert mode. During insert mode in vi, a typed character is inserted into the current buffer. In vi command mode, the same character may be used to signify some command such as moving the cursor. For example, in vi the keystroke "n" means either to insert an "n" into the buffer or to find the next occurrence in the buffer of a previously specified search string. Which of these meanings is applicable depends on if the user is currently working in insert mode.

Emacs does not have this distinction between modes. Typing "n" only means "insert 'n' into the buffer." In Emacs, therefore, all printing characters are unavailable for use as commands; escape sequences and control characters must be used instead. The default Emacs keystroke sequence for "find the next occurrence in the buffer of the previously specified search string" is ^s-return (where ^s designates Control-s). To execute this common operation in Emacs requires three keystrokes from distant parts of the keyboard instead of a single keystroke, as in vi.

In a similar way, most Emacs commands take longer to type than the corresponding vi commands. This makes the Emacs user interface considerably less efficient than vi, although some programmers are not bothered by extra typing.

To this criticism the Emacs enthusiast will respond that it is easy to modify the key bindings to make them mean anything you want. This is true, but if one gets used to an overcustomized, personal version of Emacs, the benefits of standardization are lost and you are unable to function on a new system until you have customized it.

On the positive side, Emacs has an excellent windowing facility. Under Emacs it is easy to split the display screen into two or more independent windows and use them to display several files or several parts of the same file. This feature is powerful, especially if an operating system does not provide a user interface with windows. The Sun 3 workstation I

use has a convenient way of opening new windows on the display screen, running separate tasks in them, and moving text from window to window. Although this feature is provided by the Suntools user interface software that comes with the workstation, the Emacs windows were a convenient supplement.

I found Emacs' treatment of the *newline* character to be more consistent and natural than vi's. In Emacs, the *newline* character is treated as any other; the cursor may even be positioned at it. To join two consecutive lines in Emacs it is therefore necessary only to position the cursor over the *newline* character at the end of the first line and type ^d to delete that character. In vi, a special *Join* command is required since the cursor cannot be placed at a *newline*.

After skimming through the user's guide (and armed with the reference card), I began to experiment with the system. Emacs and C-macs loaded from cartridge tape to disk with no trouble, although I was somewhat surprised to see that the program and data files (I did not have source) used over 5 MB of disk space compared to less than a quarter of a megabyte for UNIX vi. The start-up time for C-macs was also considerably greater than the start-up time for vi (18 seconds vs. 5 seconds). The 68020-based Sun is considerably more powerful than an IBM PC/AT, and I wonder what the start-up time is on an IBM PC/AT.

I had a few surprises trying the C-macs features. According to the manual, "An on-line tutorial will teach you how to use C-macs. To run this tutorial, first start C-macs as described above. Then press ^_ (control-underscore) followed by a t." When I tried this, I got an error message saying that the C-macs tutorial file was not present. According to UniPress president Mark Krieger, the company does not provide a tutorial for C-macs. A tutorial file for the underlying Emacs is present, however.

I then experimented for a while with the standard Emacs commands for cursor movement, window manipulation, string search, etc. These worked fine. I tried the *shell* command, which executes a new copy of the shell, putting its input and output into a new buffer. This command worked until I exited from the shell by typing control-d. I then got the message that the process was wedged (whatever that means) and a query as to whether I wanted to unblock it. Neither a positive nor a negative response had any effect, and the only way I could get out was to exit my log-in session via the Sun Suntools menu.

The next day another user of the system complained that the system was sluggish, and I discovered that the C-macs process was still running in the background, using large amounts of CPU. I had to kill it with a UNIX *kill* command. UniPress said this is a known bug and they are working on it.

One useful feature of Emacs is the *help* command. Typing ^_a <string> displays the names of all commands containing <*string*>. Typing ^_d <command-name> displays a short description of the named command. I typed ^_a brace and noticed that one of the commands listed was named *forward-brace*. I tried ^_d forward-brace and got the message "No help for forward-brace." I don't know why the ^_d gives help for some commands and not others.

The people at UniPress explained that descriptions are only available for the more obscure commands (the ones even experienced users would need help with). A data base with a complete set of command descriptions would take up too much room. This is a reasonable explanation, but it does not help the novice.

An interesting documented feature of C-macs allows you to compile the program in a buffer and collect the error messages automatically in another buffer. Typing ^x^n is then supposed to parse the next error message and set the cursor to the appropriate line in the source file. This is a common situation in which it is advantageous to be able to look at two files (the source file and the error file) at the same time.

I wrote a test program that contained three errors: a preprocessor error (*#defune X 1*), a compiler syntax error (*a===b;*), and a linker error (*pruntf("x");*). All three errors appeared in the buffer in turn, but the ^x^n error message parser found only one of them. If the program contained just the preprocessor error or linker error, C-macs

DON'T WASTE YOUR TIME!

Save it with BSW-Make

BSW-Make is now faster and more powerful than ever! BSW-Make speeds software development, by automating the chore of rebuilding complex software products after an editing session. No more missed compiles! No more wholesale "just in case" recompilations of the whole product! BSW-Make performs the minimum set of compilations, assemblies, and links required to correctly update your software after each edit. First time — every time. A major timesaver!

- **Now imports environment variables (MS-DOS) and logical names (VMS)**
- **Now supports .INCLUDE, .IGNORE, and .SILENT directives**
- **Macro facility now supports string substitution**
- **Syntax compatible with UNIX *make***
- **Works with any compiler, assembler, or linker**
- **For MS-DOS: only $89.95**
- **For VAX/VMS: as low as $295.00**

30-day unconditional guarantee!

Version 6 now shipping!

for free product information, call
(617) 367-6846
Ask for Department C2

The Boston Software Works, Inc.
120 Fulton Street, Boston, MA 02109

CIRCLE 51 ON READER SERVICE CARD

UNLEASH YOUR 80386!

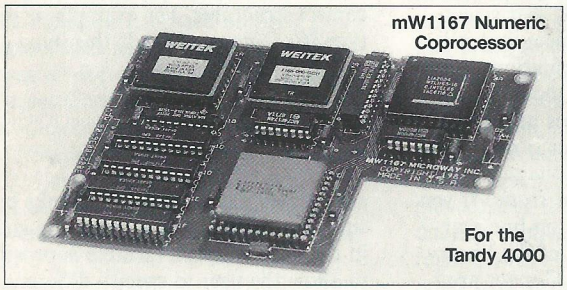
Your 80386-based PC runs at least twice as fast as your old AT. This is good, but not great. The products described below will unleash the true potential of your 80386, giving you 4 to 16 times the power of your old AT. These new MicroWay products include a family of 80386 native code compilers and the mW1167 numeric coprocessor.

Examples of the increases in capacity and performance include:

- Programs compiled with MicroWay NDP Fortran-386 execute 2 to 8 times faster than those compiled with existing 16-bit Fortrans. NDP Fortran-386 can also address up to 4 gigabytes of memory instead of the standard 640 kbytes. MicroWay's NDP compilers and the programs they generate run on MS-DOS or Unix V.
- NDP Fortran-386 generates code for the 80287, 80387 or MicroWay's mW1167. The mW1167 has a floating point throughput exceeding 2.5 megaflops, which is 4 to 5 times the throughput of an 80387 and is comparable to the speed achieved by the VAX 8600.

Equally important, whichever MicroWay product you choose, you can be assured of the same excellent pre- and post-sales support that has made MicroWay the world leader in PC numerics and high performance PC upgrades. For more information, please call the Technical Support Department at

617-746-7341



MicroWay® 80386 Support

MicroWay 80386 Compilers

NDP Fortran-386 and **NDP C-386** are globally optimizing 80386 native code compilers that support a number of Numeric Data Processors, including the 80287, 80387 and mW1167. They generate mainframe quality optimized code and are syntactically and operationally compatible to the Berkeley 4.2 Unix f77 and PCC compilers. MS-DOS specific extensions have been added where necessary to make it easy to port programs written with Microsoft C or Fortran and R/M Fortran.

The compilers are presently available in two formats: Microport Unix 5.3 or MS-DOS as extended by the Phar Lap Tools. MicroWay will port them to other 80386 operating systems such as OS/2 as the need arises and as 80386 versions become available.

The key to addressing more than 640 kbytes is the use of 32-bit integers to address arrays. NDP Fortran-386 generates 32-bit code which executes 3 to 8 times faster than the current generation of 16-bit compilers. There are three elements each of which contributes a factor of 2 to this speed increase: very efficient use of 80386 registers to store 32-bit entities, the use of inline 32-bit arithmetic instead of library calls, and a doubling in the effective utilization of the system data bus.

An example of the benefit of excellent code is a 32-bit matrix multiply. In this benchmark an NDP Fortran-386 program is run against the same program compiled with a 16-bit Fortran. Both programs were run on the same 80386 system. However, the 32-bit code ran 7.5 times faster than the 16-bit code, and 58.5 times faster than the 16-bit code executing on an IBM PC.

NDP FORTRAN-386™ **$595**
NDP C-386™ **$595**

MicroWay Numerics

The **mW1167™** is a MicroWay designed high speed numeric coprocessor that works with the 80386. It plugs into a 121 pin "Weitek" socket that is actually a super set of the 80387. This socket is available on a number of motherboards and accelerators including the AT&T 6386, Tandy 4000 and MicroWay Number Smasher 386 (Jan. '88). It combines the 64-bit Weitek 1163/64 floating point multiplier/adder with a Weitek/Intel designed "glue chip". The mW1167™ runs at 3.6 MegaWhetstones (compiled with NDP Fortran-386) which is a factor of 16 faster than an AT and 3 to 5 times faster than an 80387 .....**$1495**

Monoputer™ – The INMOS T800-20 Transputer is a 32-bit computer on a chip that features a built-in floating point coprocessor. The T800 can be used to build arbitrarily large parallel processing machines. The Monoputer comes with either the 20 MHz T800 or the T414 (a T800 without the NDP) and includes 2 megabytes of processor memory. Four or more Transputers can be easily linked together to form a Quadputer. A single T800 is comparable in speed with an mW1167-equipped 80386. The compilers to drive one or more Monoputers include Occam, C, Fortran, Pascal and Prolog.

Monoputer T414-20[1] **$1495**
Monoputer T800-20[1] **$1995**
Biputer™ T800/T414[2] **$4995**
Quadputer™ T414-20[2] **$6995**

[1]Includes Occam [2]Includes TDS

80287 ACCELERATORS

287Turbo-10 .....**$450**
287Turbo-12 .....**$550**
287TurboPlus-12 .....**$629**

80386 Multi-User Solutions

AT8™ – This intelligent serial controller is designed to handle 8 users (16 with two boards) in a Xenix or Unix environment with as little as 3% degradation in speed. It has been tested and approved by Compaq, Intel, NCR, Zenith, and the Department of Defense for use in high performance 80286 and 80386 Xenix or Unix based multi-user systems **$1299**

MicroPort Unix 5.3 is a port of the new Unix 5.3 to the 80386. MicroWay NDP-386 compilers currently run on this version of UNIX.
MicroPort Unix 5.3 .....**from $399**

PC-MOS-386™ is an 80386 operating environment that turns an AT with an AT8 into an MS-DOS multi-user system. The system makes it possible to run applications such as Lotus 1-2-3 on terminals. The operating system also has a Phar Lap compatibility mode that runs programs developed with the Phar Lap versions of MicroWay's compilers .....**from $199**

Phar Lap™ created the first tools that make it possible to develop 80386 applications which run under MS-DOS yet take advantage of the full power of the 80386. These include an 80386 monitor/loader that runs the 80386 in protected linear address mode, an assembler, linker and debugger. These tools are required for the MS-DOS version of the MicroWay NDP Compilers.
Phar Lap Tools.....**$399**

MATH COPROCESSORS

80387-16 16 MHz.....**$495**
80287-10 10 MHz.....**$349**
80287-8 8 MHz **$259**
80287-6 6 MHz **$179**
8087-2 8 MHz **$154**
8087 5 MHz **$99**

MicroWay

The World Leader in PC Numerics

P.O. Box 79, Kingston, Mass. 02364 USA (617) 746-7341
32 High St., Kingston-Upon-Thames, U.K., 01-541-5466

CIRCLE 52 ON READER SERVICE CARD

Introducing Network Version!

SEIDL VERSION MANAGER

Now SVM supports local area networks and tracks source revisions made by multiple users in both single-site and multi-site configurations.

Plus...

- Archive Database Tracks Source (and Binary) File Revisions
- Audit Trail Reporting Provides Info on Project's Development
- Revision Branches Allow Multiple Courses of Development
- Revision Merging and Deleting Provide Flexibility in Archive Maintenance
- User IDs, Privilege Settings & Passwords Help Resolve Access Conflicts and Maintain Project Integrity
- Optional Text Compression Reduces Storage Requirements
- Menu Driven Shell Makes SVM Easy to Use
- Single-Site: $299.95*
- 5-site LAN: $1000 (extendible)

Now Builds Dependencies!

SEIDL MAKE UTILITY

New program, called SMKgen, automatically constructs a dependency file by analyzing the files in a project.

Plus...

- Structured Language Used to Define Dependencies
- Rich Command Set with Over 20 Different Statements
- Ability to Handle Nested Include Files and Library Dependencies
- Performance & Functionality not Found in UNIX Make or Clones
- SMK Only: $99.95*
- SMKgen: Add $50.00

CALL TODAY

1-313-662-8086

Visa/MC/COD Accepted
Dealer Inquiries Invited

*Plus postage and Handling

SEIDL COMPUTER ENGINEERING

3106 Hilltop Dr., Ann Arbor, MI 48103

CIRCLE 53 ON READER SERVICE CARD

would report "No Errors," even though the error message was in the buffer. To catch these different error messages it is necessary to modify the error-parsing algorithms. The extensibility of Emacs allows this to be done without source code.

Another feature of C-macs is the ^x^d keystroke sequence that prints one-line and longer descriptions of common C keywords. One need only put the cursor at the word to be described and type ^x^d. I put the cursor at an occurrence of the token *char* and got a nice description of the meaning of the keyword in C. Likewise, I got descriptions of the words *int, if, switch,* and all the library routines I tried, but was unable to get any help on the terms *ifdef, void, register, static, struct, case, default, break,* and *sizeof.*

The data base is incomplete, but it is easy for the user to add definitions to it. The explanation that only the more obscure terms are defined does not hold water here; the programmer is more likely to know the meaning of the terms *int* or *if* than the terms *void* or *static.* If you have a UNIX system with on-line manual pages, the descriptions of the library routines available through C-macs are redundant.

C-macs gives the user the option of executing some common commands via a menu rather than typing the commands. The keystroke sequence to activate the menu is Escape-m. I was confused because sometimes typing Escape-m produced a menu while at other times it did nothing. I finally noticed that for Escape-m to produce a menu, the mouse on the Sun must be touched after entering C-macs. Without moving the mouse cursor at least a fraction of an inch, the menu will not activate. I checked with UniPress on this problem and they could not duplicate it; either it was some peculiarity of my workstation or it had been corrected in the most recent version of the Emacs software.

One more peculiarity associated with menus occurs with windows. Ordinarily, moving the mouse cursor to a Suntools window will activate that window. Thus it should be possible to have one Suntools window running Emacs, another running a shell or some application program, another running vi, etc. Whenever a menu is up in an Emacs window, however, other Suntools windows not running Emacs cannot be activated. I have run many programs on the Sun under Suntools and have never seen one of these programs interfere with others in different windows in this way.

UniPress called this drawback a feature instead of a bug, but the explanation was not too clear to me. In any case, once the user is aware of the problem, it is a simple matter to kill the menu before moving to another window.

Finally, the statement in the newsletter that "all menu items can be bound to keys, and these bindings are shown in the menus" is incorrect. The key bindings are not shown in the menus.

One of the more confusing aspects of C is the C type declarator. C-macs provides a feature that allows the user to set the cursor at a type declarator and get an English-language description of the meaning of that declarator. For example, if you set the cursor at the declaration *int (*f[])();* C-macs will tell you: "declare *f* as array of pointer to function returning *int.*" The facility that does this is a public-domain C program called cdecl that is supplied with C-macs.

Another feature furnished by C-macs is a set of macros that expand into C control structures. For example, if you type *if(,* C-macs expands this into:

```
if() {

}
```

and allows you to fill in the blanks by positioning the cursor and inserting strings in the usual way. A variable is provided to allow the user to specify different styles such as:

```
if()
{

}
```

or:

```
if()
    {

    }
```

Unfortunately, not all of the expansions noted in the documentation work. For example, typing *elseif* is supposed to produce:

```
else if (){

}
```

but doesn't. There are also some undocumented expansions. For example, typing:

```
main(
```

expands to

```
main()
{

}
```

and typing the characters *argc* after the opening parenthesis immediately produces:

```
main(argc, argv)
int argc;
```

```
int argc;
char *argv[];
{

}
```

This is very convenient. Unfortunately, if the C-macs variable *elec-c-auto-close-paren* is set (which is always supposed to create parentheses and brackets in matching pairs), these expansions do not work correctly.

Some programmers write vi macros to produce keyword expansions similar to those just displayed, but the C-macs facility is somewhat more sophisticated than what can be done in vi; for example, C-macs will suppress expansion within a comment.

Support is a critical consideration in selecting software. I did not have the Emacs support option; nevertheless, when I called UniPress to discuss the minor problems I had, company representatives were very helpful. Their knowledgeable support person listened patiently to all my questions, offered suggestions and explanations, and took bug reports when necessary. (UniPress did not know I was reviewing the product.)

The basic idea behind C-macs is a good one: to provide an editing tool with built-in knowledge of the C programming language. The Emacs editor upon which C-macs is built is basically a sound one, although it seems to have some problems interacting with the Suntools windowing system. The organization and presentation of the Emacs/C-macs documentation is just right and serves both the beginner and the expert well.

I was somewhat disappointed by the many minor inconsistencies and omissions. The testing and quality assurance for the product should have caught more of these problems. However, since the product has been on the market less than a year, I would expect the quality to improve. I was very pleased with the level of telephone support.

If you know and like Emacs and do a lot of programming in C, C-macs may be for you. You need to decide whether the documentation, support, and C language extensions make the $645+ for C-macs worthwhile compared to the several freeware versions of Emacs. If you are currently a vi user and are considering C-macs, you also need to consider the differences between vi (small and efficient) and Emacs (big and extensible).

What the world is really waiting for is an incrementally compiling syntax-directed editor, a unified editor/compiler that produces only syntactically correct programs. UniPress C-macs is a small step in this direction.

By Alan Filipski

The Advanced Programmer's Editor That Doesn't Waste Your Time

For DOS, Microport UNIX, or . . . OS/2 Protected Mode

EPSILON

- Fast, EMACS-style commands—completely reconfigurable
- Run other programs without stopping Epsilon—concurrently!
- C Language support—fix errors while your compiler runs
- Powerful extension language
- Great on-line help system
- Multiple windows, files
- Regular Expression search
- Unlimited file size, line length
- Supports large displays
- 30 day money-back guarantee
- Not copy protected

Only $195

Lugaru Software Ltd.

5740 Darlington Road
Pittsburgh, PA 15217

Call (412) 421-5911

for IBM PC/XT/AT's or compatibles

CIRCLE 55 ON READER SERVICE CARD

Personal REXX
for the IBM PC

- ★ Interpreter for the full **REXX** language, including all of the standard **REXX** instructions, operators, and built-in functions
- ★ Sophisticated string manipulation capabilities
- ★ Unlimited precision arithmetic
- ★ Direct execution of DOS commands from **REXX** programs
- ★ Built-in functions for DOS file I/O, directory access, screen and keyboard communication, and many other PC services
- ★ Compatible with VM/CMS version of **REXX**
- ★ Uses include:
  - — Command programming language for DOS
  - — Macro language for the **KEDIT** text editor
  - — Can be interfaced by application developers with other DOS applications, written in almost any language

Mansfield Software Group, Inc.
P.O. Box 532
Storrs, CT 06268
(203) 429-8402

$125 plus $3 shipping
MC, VISA, AMEX, COD, PO, CHECK

CIRCLE 54 ON READER SERVICE CARD

COMBINE THE
RAW POWER OF FORTH
WITH THE CONVENIENCE
OF CONVENTIONAL LANGUAGES

HS/FORTH

Yes, Forth gives you total control of your computer, but only HS/FORTH gives you implemented functionality so you aren't left hanging with "great possibilities" (and lots of work!) With over 1500 functions you are almost done before you start!

WELCOME TO HS/FORTH, where megabyte programs compile at 10,000 lines per minute, and execute faster than ones built in 64k limited systems. Then use AUTOOPT to reach within a few percent of full assembler performance — not a native code compiler linking only simple code primitives, but a full recursive descent optimizer with almost all of HS/FORTH as a useable resource. Both optimizer and assembler can create independent programs as well as code primitives. The metacompiler creates threaded systems from a few hundred bytes to as large as required, and can produce ANY threading scheme. And the entire system can be saved, sealed, or turnkeyed for distribution either on disk or in ROM (with or without BIOS).

HS/FORTH is a first class application development and implementation system. You can exploit all of DOS (commands, functions, even shelled programs) and link to .OBJ and .LIB files meant for other languages *interactively!*

I/O is easier than in Pascal or Basic, but much more powerful — whether you need parsing, formatting, or random access. Send display output through DOS, BIOS, or direct to video memory. Windows organize both text and graphics display, and greatly enhance both time slice and round robin multitasking utilities. Math facilities include both software and hardware floating point plus an 18 digit integer (finance) extension and fast arrays for all data types. Hardware floating point covers the full range of trig, hyper and transcendental math including complex.

Undeniably the most flexible & complete Forth system available, at any price, with no expensive extras to buy later. Compiles 79 & 83 standard programs. Distribute metacompiled tools, or turnkeyed applications royalty free.

HS/FORTH (complete system):	$395.
HS/FORTH: Tutorial & Ref (500 pg)	$ 59.
Forth: Text & Reference (500 pg)	$ 22.
HS/FORTH Glossary	$ 10.
GIGAFORTH Option (Beta release)	$245.

(Native Mode from SOFTMILLS, INC.)

VISA Visa Mastercard MasterCard

HARVARD
SOFTWORKS
PO BOX 69
SPRINGBORO, OH 45066
(513) 748-0390

CIRCLE 56 ON READER SERVICE CARD

OPAL

System requirements: IBM PC or compatible, MS-DOS, 192K RAM
Price: $99; OPART screen painter—$40; large memory model—$25; math option—$40; cross-reference tool—$40; run-time system—$99
Available from: The Software Factory, 15301 Dallas Pkwy., Ste. 750, LB44, Dallas, Texas 75248, (214) 490-0835
Support: Telephone support, not copy protected

OPAL is advertised as a "batch executive language that gives you the control you need over the PC-DOS/MS-DOS environment." While this is true, OPAL is also an interpreter and prototyping aid with features to aid in entering data, defining custom menus, and developing your own shell.

The small-memory model (the default) handles programs that use up to 64K of data while the large-memory model uses all of the system memory. A setup file lets the OPAL user to control *trace, traceback,* and *fastio.* While OPAL is slow as an interpreter, a tokenizing compiler can be purchased for enhanced speed. (On some benchmarks the interpreter execution time was about 23 seconds, while the compiler time was 9 seconds.)

OPAL is delivered with several examples that show the user how to write OPAL programs, define and use menus, and create a custom shell that gives DOS a completely different feel. One of this shell's unusual features is that it does not replace COMMAND.COM but builds upon it. Commands are processed by the shell if it recognizes them, while unknown commands are simply handed to the DOS command interpreter. This feature permits renaming commands or removing them entirely.

As a programming language, OPAL has some of the characteristics of BASIC and C. Assigning a variable *X* the value of 1 is done by the statement *LET X = 1.* The *file open* function requires an argument of "r" or "w" (quotes included) to specify read or write mode. The BASIC-like *LET* is specified in the OPAL manual but seems to be optional in the interpreter (as in BASIC), but the compiler demands it.

An interesting fact about OPAL variable assignments is that the variable type is determined at run time. This means that in the statements *LET X = 1* and *LET X = 1.0* the variable *X* is first an integer and then a real number. This type of conversion is automatic and could potentially produce a problem since *LET X = 1/2* and *LET X = 1.0/2* give answers of 0 and 0.5. An OPAL programmer must be aware of this distinction while writing so subtle errors can be prevented.

In addition to *LET,* OPAL has *BREAK, CALL, CONTINUE, DATA, DECLARE, DEFINE, DO-END, DOS, GLOBAL, IF-THEN-ELSE, PRINT, PROC, REMOVE, RETURN, SELECT,* and *SHOW* statements.

The *BREAK* statement exits a loop created by a *DO-END* pair and accepts an optional parameter that is used with nested loops to control the level at which thebreak takes place. If no level is specified, the default is the lowest level.

The *CALL* statement comes in three varieties: *CALL OPAL* to call another OPAL program, *CALL SYSTEM* to execute an MS-DOS/PC-DOS internal command, and *CALL EXEC* to execute operating system external commands (.COM, .EXE, and .BAT files). For the *CALL OPAL* variant, the number of nested calls is limited to the lesser of the number of open files specified in CONFIG.SYS or 64. (Since five files are used by OPAL internally, the actual number is 57.) *CALL SYSTEM* loads the command processor and passes a command line with arguments to COMMAND .COM for processing, while *CALL EXEC* loads a program into memory and executes it.

OPAL may be overlaid by the child process created with *CALL EXEC* if there is not sufficient memory to permit both processes to be resident (controlled by a Boolean argument in the call). If OPAL is overlaid, control returns to DOS after the program is executed. If OPAL is not overlaid, control is returned to OPAL. If the global variable *SAVE_* is true, the screen is returned to its *pre-call* state.

The *CONTINUE* statement is executed conditionally to transfer control to the bottom of a *DO-END* block. The difference between *BREAK* and *CONTINUE* is that *BREAK* transfers control just past the *END* statement while *CONTINUE* transfers control just before the *END. Thus, BREAK* exits the loop while *CONTINUE* goes to the bottom of the loop, terminating the current iteration and starting the next one.

DATA is an output statement used to display a single variable type on the screen. *DATA* statements are confined to a *DO DATA* block I will describe later.

The *DECLARE* statement is used to allocate space for an array and optionally initialize it. An array can also be created by an initial reference to the largest element. Thus, *LET X[20] = 1.5* creates an array of 20 elements even if *X* has not

been formally declared. With this usage, the first reference must be to the upper bound of the array. The *DECLARE* initialization of an array is faster than using a *DO* loop.

The *DEFINE* statement is the workhorse for producing forms and menus with OPAL. Both forms and menus start with a *DEFINE* that lists the form or menu name and whether it is a form or a menu. Following this is a *BORDER* definition that specifies a position for the heading (left justified, centered, or right justified), heading string, border type, width, and height. Next is a *LABEL* with placement and text string arguments.

For a menu, a *CHOICE* option then indicates a number to return if this choice is selected, location of the choice in the border, a text string describing the choice, cursor movement information for moving around the menu, and the name of an optional help file to display if the user requests it. The choice substatement is repeated as needed up to 50 times.

Forms have *FIELD* statements (rather than *CHOICE*s) that denote placement of the field, a field number, type and length of the field, edit action, variable, help file, and value verification information to ensure that the user enters a valid value. The edit action permits the programmer to force case to upper or lower, justify, zero-fill numeric fields, and protect fields.

Both the form and the menu versions of *DEFINE* are terminated with *END*. For the user who depends heavily on menus and/or forms, a companion product, OPART, speeds up the generation of the code to handle these features by using functions keys to save typing in the information.

The *DO* statement comes in eight flavors. The simple *DO* does not form a loop but is used (with an *END* statement) to make compound statements, which is analogous to the *BEGIN-END* block in Pascal. The *DO* statement can be used to perform loops with *DO WHILE, DO UNTIL,* an incremental *DO (DO Y=1 TO 10 [BY step])*, a combination of incremental and conditional (*DO Y=1 BY 1 WHILE Y<100)*, incremental with a *NEXT* clause (*DO Z=1.001 NEXT Z*2 UNTIL Z>3)*, *DO IN* to test for set membership, and the *DO DATA* statement to print information on the screen with the *DATA* statement. An example of a *DATA* output loop is:

```
X=4.5
DO fn="CON" DATA WHILE X>5
X=X*1.02
DATA X
END
```

The *DOS* statement is used to leave OPAL and return to the operating system. A value for the system variable *ERRORLEVEL* may be specified to flag abnormal termination if OPAL is run from a batch file. The *GLOBAL* statement is used to make a local variable a global variable so it can be referenced in any routine. The *IF-THEN-ELSE* statement is used to control program flow and is used like the Pascal equivalent.

The *PRINT* statement is a more general output statement than the *DATA* statement and is not confined to a *DO-DATA* block. *PRINT* is almost identical to *PRINTF* in C except that the new line character is %n rather than \n. Executing the statements

```
LET age = 37
LET name = "George"
PRINT("%s is %d years
old.%n",name,age)
```

results in the the following line:

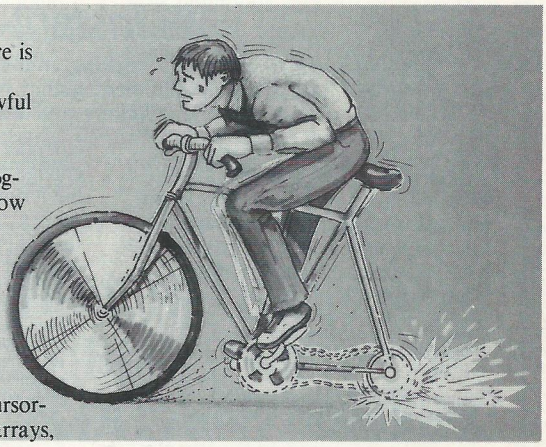
George is 37 years old.

If another %n were placed after the first, a blank line would also be printed.

If You Have Turbo C You Have Half Your C-Programming Vehicle

Turbo C is a great compiler but there is one vital cog missing—debugging. Without it, you have to spend an awful lot of energy to go a short distance.

Gimpel Software's C-terp, long recognized as the leading C interpreter, now fully supports Turbo C with complete compatibility guaranteed.



The missing wheel that will turn your half-cycle into a bicycle

Interactive Debugger—Our debugging facilities include split screen (code in upper portion, dialog in lower), breakpoints (sticky, temporary, line/function, cursor-directed), display of structures and arrays, execution of any expression (even those involving macros), function traceback with arguments, watch expressions and watch conditions (watchpoints). Our watch expressions can be structs or arrays. We catch out-of-bounds pointers!

No Toy—Full K&R with ANSI enhancements. Multiple-module with a built-in automatic make. It has virtual memory option (with optional direct use of extended memory) and a shared symbol option for those big programs. It supports graphics, dual displays and the EGA 43-line mode.

Links to external libraries—(both code and data, automatically) which can call back to interpreted functions. Function pointers are compiler compatible.

100% Turbo-C compatible.—Same header (.h) files, data alignment, bit field orderings and preprocessor variables as your compiler. We link in your compiler's library.

Our reconfigurable editor—is multifile and comes with a configuration script to mimic Turbo's editor.

C-terp

Order C-terp today!

Call (215) 584-4261

Introductory Price for Turbo C-terp:
$139.00

VISA, MC, COD—30 day money back guarantee

C-terp Version 3.0 is also available for the following compilers:
Microsoft, Lattice, Aztec, C86, and Mark Williams ($298) and Xenix ($498).

GIMPEL SOFTWARE
3207 Hogarth Lane
Collegeville, PA 19426

C-terp is a trademark of Gimpel Software, and Turbo C of Borland International.

The *PROC* statement is used to establish an entry point for a routine that will be *CALL*ed externally. It is used in the same fashion as a *PROCEDURE* statement in Pascal or a *SUBROUTINE* statement in FORTRAN, and lists the parameters that may be passed to the routine when it is called.

The *REMOVE* statement is used to remove variables from the symbol table. More than one variable may be removed at a time but only variables from the current level may be removed if local routines are used.

The *RETURN* statement leaves the current program and returns to the calling program while passing back an optional return value.

SELECT is a branch similar to the *SWITCH* statement in C. The syntax is:

```
SELECT original_expression
   WHEN check_expression statement
{WHEN check—expression statement}
{OTHERWISE statement}
```

The optional *OTHERWISE* clause is executed if no match is found on any of the *WHEN*s. If the same statement is to be executed for multiple check expressions, it may be separated by commas following a single *WHEN*.

The *SHOW* statement is an output statement similar to *PRINT* except that it accepts a single argument only. This may be a string constructed by the *STRFMT* (string format) function or a variable.

OPAL supports logical, relational, and arithmetic functions and tests. The categories of functions available in OPAL are string, numeric, screen, file, disk/directory, date and time, list and element, and system functions.

The string functions provide OPAL users with a method to format a string for output, fill a string, locate a character in a string, convert character case, extract a substring, find string length, parse a line, or translate a string. During string conversion, the programmer may specify arguments to control discarding or compressing of white space, prevent alteration of characters in quotes, and eliminate nonprinting characters.

A function in the string group, *STRAT,* positions the cursor at a specified row and column to control the location of printed data on the screen. The string format function, *STRFMT,* accepts the same arguments as the *PRINT* function (which outputs to the screen only) and is used to output formatted data to a file with the *FIWRITE* function.

The numeric functions are a bit sparse; only *ABS, EVEN, INT, MAX, MIN, ODD,* and *ROUND* functions are available with the basic product. An OPAL math option adds trigonometric functions, exponentials, logs (natural and base 10), square root, and miscellaneous functions. The miscellaneous functions include *CEIL* (returns the smallest integer greater than or equal to the argument), *FLOOR* (returns the largest integer less than or equal to the argument), *FMOD* (computes the floating-point modulus), and *MODF* (splits a floating-point value into integer and decimal portions).

The screen functions clear the screen, place a graphic border on the screen, scroll a region on the screen, save a screen image, display a menu or form, save or load a menu or form to/from a file, modify a menu choice or form field, and output a help file to the screen.

The console functions provide the ability to flush the keyboard stack, get a character, check for a keyboard hit, wait for either a keystroke or an elapsed time, get data with a prompt, and put a character to the console. The *CONPROMPT* function reads from the console with a prompt, checks the entry for validity by checking range, membership in a set, and comparison to a maximum or minimum value, and returns a default value if the user enters a carriage return only.

The file functions enable the programmer to open and close files, check for the existence of a file, delete a file, get the

Complete C Programs in Half the Time, with *Instant-C*™

***Instant*-C helps you create a working, well-tested program *faster* than any other interactive C development system. Much faster than traditional compilers, linkers, and debuggers.**

Hard to believe? Here's how we do it. Because *Instant-C* is a high-performance interpreter there are **no compile or link delays.** Change your program, then test it immediately. No matter how large your program, the turnaround time is just seconds.

"Instant-C means instant gratification."—*PC Magazine,* **Editor's Choice** for best C interpreter. *10/29/85*

"Time after time, the Instant-C prompt was starting back just barely after pressing Enter."—*PC Tech Journal, 5/86*

Source-level debugging saves your time.

- set any number of conditional breakpoints in your program;
- stop execution from keyboard;
- single-step by source statement;
- examine and change variables or code, and continue execution;
- execute any statement or function directly for instant testing;
- display source code back-traces;
- source code animation;
- monitor data changes;
- full-screen and multi-screen support, even with non-standard graphics devices.

"The resulting debugging and testing capabilities are fantastic and the detailed trace/debug/display commands make it easy."—*The C Journal, 5/85*

Run-time checking stops your program as soon as errors occur, when bugs are easiest to understand and fix.

- pointer references checked for reasonableness;
- array indexes checked within declared bounds.

Not only does *Instant-C* help you quickly change, test, check and debug your code, but it runs your program faster than any other C interpreter. **Fifty to 500 times faster!** Fast enough for *real* programs, even fast enough for real-time programs.

"It is much faster than any of the other products mentioned and was the only one able to complete the standard SIEVE in a reasonable time. Clearly, this high speed allows much more complex problems to be attacked with Instant-C than with any of the other products discussed."—*Computer Language, 2/86*

Immediate feedback and more than 400 diagnostics makes *Instant-C* great for learning C. Full K&R and the ability to **link compiled object code and libraries** (Lattice and Microsoft) makes *Instant-C* compatible with your existing programs.

"When you get right down to it, I don't think there's a better way you could learn C."—*Programmer's Journal, 3/85*

"Clearly, Instant-C is the performance champion."—*PC Tech Journal, 5/86*

The bottom line for your business is increased productivity. The result for you is a job well done, and quickly.

"We sincerely feel that Instant-C can have a major positive impact on programmer productivity."—*Computer Language, 2/85*

"Instant-C by Rational Systems is a C programmer's dream."—*Micro/Systems Journal, 3/86*

Version 2 is available for MS-DOS and PC-DOS, and comes with a full 31 day **money back guarantee.** *Instant-C* is only $495. Order today! Call or write for full information.

Rational Systems, Inc.
P.O. Box 480
Natick, MA 01760
(617) 653-6194

CIRCLE 57 ON READER SERVICE CARD

SAS Institute Inc. Announces

Lattice C Compilers for Your IBM Mainframe

Two years ago...

SAS Institute launched an effort to develop a subset of the SAS® Software System for the IBM Personal Computer. After careful study, we agreed that C was the programming language of choice. And that the Lattice® C compiler offered the quality, speed, and efficiency we needed.

One year ago...

Development had progressed so well that we expanded our efforts to include the entire SAS System on a PC, written in C. And to insure that the language, syntax, and commands would be identical across all operating systems, we decided that all future versions of the SAS System—regardless of hardware—would be derived from the same source code written in C. That meant that we needed a C compiler for IBM 370 mainframes. And it had to be good, since all our software products would depend on it.

So we approached Lattice, Inc. and asked if we could implement a version of the Lattice C compiler for IBM mainframes. With Lattice, Inc.'s agreement, development began and progressed rapidly.

Today...

Our efforts are complete—we have a first-rate IBM 370 C compiler. And we are pleased to offer this development tool to you. Now you can write in a single language that is source code compatible with your IBM mainframe and your IBM PC. We have faithfully implemented not only the language, but also the supporting library and environment.

Features of the Lattice C compiler for the 370 include:

- **Generation of reentrant object code.** Reentrancy allows many users to share the same code. Reentrancy is not an easy feature to achieve on the 370, especially if you use non-constant external variables, but we did it.
- **Optimization of the generated code.** We know the 370 instruction set and the various 370 operating environments. We have over 100 staff years of assembler language systems experience on our development team.
- **Generated code executable in both 24-bit and 31-bit addressing modes.** You can run compiled programs above the 16 megabyte line in MVS/XA.
- **Generated code identical for OS and CMS operating systems.** You can move modules between MVS and CMS without even recompiling.
- **Complete libraries.** We have implemented all the library routines described by Kernighan and Ritchie (the informal C standard), and all the library routines supported by Lattice (except operating system dependent routines), plus extensions for dealing with 370 operating environments directly. Especially significant is our byte-addressable Unix®-style I/O access method.
- **Built-in functions.** Many of the traditional string handling functions are available as built-in functions, generating in-line machine code rather than function calls. Your call to move a string can result in just one MVC instruction rather than a function call and a loop.

In addition to mainframe software development, you can also use our new cross-compiler to develop PC software on your IBM mainframe. With our cross-compiler, you can compile Lattice C programs on your mainframe and generate object code ready to download to your PC.

With the cross-compiler, we also offer PLINK86™ and PLIB86™ by Phoenix Software Associates Ltd. The Phoenix link-editor and library management facility can bind several compiled programs on the mainframe and download immediately executable modules to your PC.

Tomorrow...

We believe that the C language offers the SAS System the path to true portability and maintainability. And we believe that other companies will make similar strategic decisions about C. Already, C is taught in most college computer science curriculums, and is replacing older languages in many. And almost every computer introduced to the market now has a C compiler.

C, the language of choice...

C supports structured programming with superior control features for conditionals, iteration, and case selection. C is good for data structures, with its elegant implementation of structures and pointers. C is conducive to portable coding. It is simple to adjust for the size differences of data elements on different machines.

Continuous support...

At SAS Institute, we support all our products. You license them annually; we support them continuously. You get updates at no additional charge. We have a continuing commitment to make our compiler better and better. We have the ultimate incentive—all our software products depend on it.

For more information...

Complete and mail the coupon today. Because we've got the development tool for your tomorrow.

SAS®

SAS Institute Inc.
SAS Circle, Box 8000
Cary, NC 27511-8000
Telephone (919) 467-8000 x 7000

I want to learn more about:

- ☐ the C compiler for MVS software developers
- ☐ the C compiler for CMS software developers
- ☐ the cross-compiler with PLINK86 and PLIB86

today...so I'll be ready for tomorrow.

Please complete or attach your business card.

Name _____

Title _____

Company _____

Address _____

City _____ State _____ ZIP _____

Telephone _____

Mail to: SAS Institute Inc., Attn: CC, SAS Circle, Box 8000, Cary, NC, USA. 27511-8000. Telephone (919) 467-8000, x 7000

CL 11/87

SAS is the registered trademark of SAS Institute Inc., Cary, NC, USA. Lattice is the registered trademark of Lattice, Inc. PLINK86 and PLIB86 are trademarks of Phoenix Software Associates Ltd. UNIX is the trademark of AT&T. Copyright © 1985 by SAS Institute Inc. Printed in the USA.

file characteristics (attributes, size, and date and time stamp), set the file characteristics, rename a file, read or write a file, and find the first matching directory entry. When a file is opened it has an access mode to control whether it is in read, write, append, or update mode. Files can be opened in either text or binary mode.

The disk/directory functions permit changing the current directory or path name, getting free space, getting the drive or path name, and making or removing a directory. The date and time functions calculate the difference between two dates and times, calculate the day of week, make a date/time stamp, get the current time and date from the system, and split the time and date into separate variables.

The list and element functions allow adding or inserting elements in a list, deleting an element, returning or updating the current element, returning the data type of the current element, positioning the pointer, testing for an empty list, testing for the head or tail of a list, deleting an entire list, creating a list, making a list of directory names, getting a list from or writing the list to a file, and returning the number of elements in the list.

The system function group adds the ability to get the latest system error number and/or text, get environment variables, find the amount of free memory, set an environment variable, generate a tone, and suspend program execution.

OPAL provides the ability to create powerful, intelligent batch files and should be attractive to the user who can't get enough power out of DOS. Users who have a minicomputer background and are accustomed to a more powerful command language (such as DCL on the VAX) will appreciate the versatility of OPAL. OPAL provides a quick-to-code, easy-to-maintain custom interface for users who need insulation from DOS. Finally, OPAL can be used to prototype an application with reasonable speed and effort before the actual coding begins and changes become difficult to effect.

The OPAL documentation is good and consists of a reference manual plus numerous examples. The manual is not a tutorial and I recommend using the examples as an introduction to using OPAL effectively. The examples on the disk worked without flaw, but I found some errors in the examples in the manual.

The telephone support from The Software Factory was helpful and courteous. I reported several errors in the manual but the Software Factory was already aware of them. The manual I received was marked v. 1.1. Since the company is shipping v. 1.3 the new manual should not have these errors.

If I could change one thing about OPAL, I would improve the file I/O capability. The only way I was able to perform a formatted write to a file when running benchmarks comparing OPAL to BASIC was to create a string with the *STRFMT* function and then write it to a file with a *DATA* statement. This capability is too important to require two steps. A file equivalent of the *PRINT* statement should be provided.

On the whole, I feel that OPAL is a useful, reasonably priced product. For those who have one of the needs it fulfills, OPAL will be a valuable tool.

Update: A new release of OPAL, v. 1.3, now being shipped, provides an environment block for transmitting information between applications, additional bitwise operators, file-handling functions, and other features. The Software Factory claims the new version is 5%-25% faster while running applications interpretively.

By George Grimes

LEAVE DEVELOPMENT TROUBLES BEHIND

With Realia's development tools, you can build efficient mainframe applications right on your PC. Every stage of development gets easier, whether you're maintaining an existing application or creating a new one.

- Realia COBOL, the fastest PC COBOL compiler, produces the most efficient executable programs.
- RealCICS® lets you develop and run mainframe CICS programs on the PC.
- RealDBUG offers interactive, source-level debugging better than any mainframe tool.

Choose Realia, and your applications backlog can soon be just a memory.

REALIA®

10 South Riverside Plaza, Chicago, IL 60606 • (312) 346-0642 • Telex 332979

CIRCLE 58 ON READER SERVICE CARD

Command Plus

System requirements: IBM PC/XT/AT or compatible
Price: $79.95
Available from: ESP Software Systems Inc., 11965 Venice Blvd., Ste. 309, Los Angeles, Calif. 90066, (213) 390-7408, (800) 992-4ESP (outside Calif.)
Support: Technical support available

Command Plus is a group of programs designed to provide enhanced and extended operating system services beyond those provided by PC-DOS or MS-DOS. CP/M hackers will appreciate the loose analogy: Command Plus is to MS-DOS as ZCPR3 is to CP/M. Command Plus is also a good illustration of the saying "there ain't no such thing as a free lunch." The added capabilities have their price in disk space and RAM use. First I will describe Command Plus's features and capabilities; then I'll discuss the price of lunch.

The central program in the Command Plus system is a replacement for COMMAND.COM. Some of the DOS internal commands have been turned into external programs, including the greatly enhanced batch processor and a text file browser program.

I have divided the features of Command Plus into three categories: enhancements to existing DOS features, new functions, and the advanced batch processor.

Command Plus adds many options to the *DIR, COPY* and *DEL* commands, including the ability to execute on multiple files; *MKDIR, REN,* and *TYPE* also acquire this capability. *CLS* (clear screen) accepts an attribute specifier and the *TIME* and *DATE* commands have a display-only mode.

For the most part, Command Plus command syntax and actions are compatible with those of regular DOS commands, but some notable exceptions should be explained. (I should also note that the *DIR, COPY,* and *DEL* commands are no longer resident as a part of the command processor but are programs that execute from disk.)

Almost all Command Plus commands accept multiple file names as command arguments. For example, *DIR *.COM *.EXE* would list only the files with *COM* and *EXE* extensions on the default drive. This feature extends to commands such as *TYPE* as well, so *TYPE B:*.BAT* would type all of the BAT files on drive B:, one after another.

DIR, COPY, and *DEL* have a multitude of options. The *DIR.EXE* program can sort directory listings (forward or reverse) by the file name, extension, date, time, and size; you can even specify minor sorting sequences. For example, *DIR /Oesn* would display the current directory sorted first by file extension, then by file size within extension, and finally by file name.

The default display format for the *DIR* command is the same as the DOS */W* (wide) option and is also unsorted unless you specify a sort order. Specifying a sort order gives you the familiar format of a DOS directory display. Command Plus's *DIR* has the ability to exclude normal files and subdirectories from the listing and include hidden files and system files, which are not normally displayed.

ESP Software Systems' thoughtful inclusion of a help feature in a utility program with this many features is a definite plus. On the other side of the ledger, I wish that more information were available in the directory listings. For example, the wide listing could include the file size rounded to the nearest K for each file. It would also be handy to show the number of bytes used by the files as well as the number of bytes free, which is displayed exactly the same way as in DOS.

Now COBOL Programmers Can Do Formatted Screens Quickly and Easily - with SCREENIO.

SCREENIO is the Screen Manager for Realia™ COBOL.

SCREENIO is a high-performance, COBOL-specific screen manager for the Personal Computer and compatibles. It's powerful, offers an impressive array of features, yet is easy to use. SCREENIO was written by experienced COBOL professionals and represents a truly COBOL approach to screen management on the PC.

Design and Revise Screens Painlessly With Our Panel Editor.

Use our interactive Panel Editor Facility to design your screens with a What-You-See-Is-What-You-Get approach. You can easily modify and experiment with screen layouts. Type titles and descriptive information on the screen, draw lines and boxes using the cursor movement keys, and paint the colors you want using the cursor keys or block functions. Use all 256 color combinations on your screens.

Define your data fields in COBOL terms; use your own data name, PICTURE and VALUE clauses. Specify the field type (various alpha and numeric types, hex, etc.), valid entries for the field (if you want SCREENIO to edit your data for you), and various other items. The Panel Editor will produce a copybook that contains this information. Panel revisions rarely mean changing your source code.

Programming for SCREENIO is Easy.

You can display your screen and accept data with as few as two statements; a COPY *panelname* in your WORKING-STORAGE SECTION, and a CALL SCREENIO statement in the PROCEDURE DIVISION. SCREENIO is linked with your application just as any other COBOL subroutine would be.

SCREENIO takes care of tabbing, data validation, and other functions expected of a good screen manager. When we return to your COBOL program, the data from the user fields on the screen is in your variables.

Yes, we've always done windows. SCREENIO supports Edit Masks, Automatic Error Detection and Editing, High Speed Video, Hot-Fields, Key Redefinition, Foreign Language Capability, Custom Tabbing Order, Color Control at Runtime, PC Speaker Control, and much, much more. A BMS Mapset Import is available. Most programmers are amazed at how easy it is to build flashy applications using SCREENIO and COBOL.

Our Support is Outstanding.

We provide superb telephone support. Upgrades are distributed to licensed users automatically for the first year. There are no Runtime Fees - the code you develop is yours to distribute freely.

Call us to discuss your application. Get a feel for the kind of quality support we provide.

We'll Send You a Free Demonstration Diskette.

This limited version of our Panel Editor shows how you design panels. Because it's written with COBOL and SCREENIO, it also shows the kind of features and performance you can expect in your applications.

SCREENIO 2.1 is only $400 plus shipping. Save $100 - order SCREENIO and Realia COBOL from us. We also offer packages with Kedit, a great programming editor. Ask about our other programming tools, too.

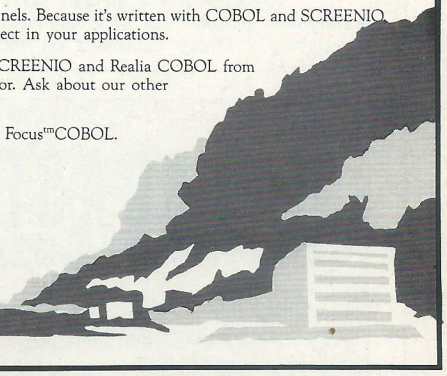
Not satisfied with FORMS™? Ask about SCREENIO for Micro Focus™ COBOL.

In a hurry? Most orders are *delivered* within 24 hours!

- Amex MC Visa Check -

NORCOM

Northern Computing Consultants
P.O. Box 020897 • Juneau, Alaska 99802
(907) 780-6464 • TELEX 510 601 4951 (Norcom)



NOT COPY PROTECTED

CIRCLE 59 ON READER SERVICE CARD

Introducing

DATABASE Programming & Design

the first magazine to focus on practical, technical solutions for DBMS specialists

—from the publishers of AI Expert, Computer Language, and Unix Review

Now, for the first time, MIS/DP managers, database administrators, designers, analysts and other technical managers will have one information source that concentrates on the technical side of database management systems. Written for technical specialists, DATABASE PROGRAMMING & DESIGN is geared to answer your most pressing design and programming questions. In *technical depth.*

Our editors know that DBMS people face big challenges. How to meet, design and manage corporate information requirements. How to cut down on the applications backlog. How to increase productivity of DP systems and end users. Designing custom applications. Evaluating, choosing and using software products specific to DBMS needs.

DATABASE PROGRAMMING & DESIGN, every month, will help you meet these technical challenges. By concentrating on hard, usable DBMS-specific information you need, our editors will save you hours and hours of time from trial-and-error mistakes. They'll give you the help you need with your management and technical problems..

Every monthly issue will bring you practical, helpful topics like these:

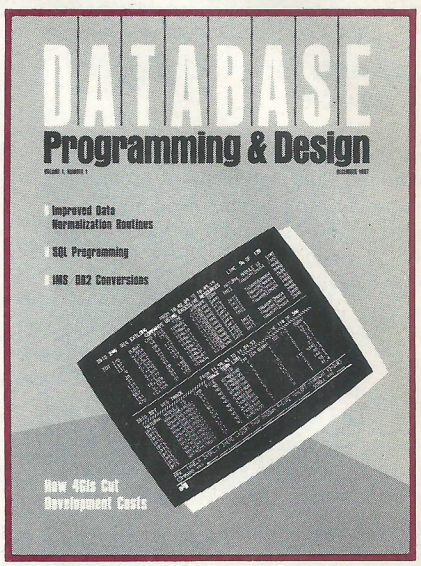
• DB2 support tools • distributed query processing • location transparency • operating system independence • normalization • conceptual data modeling • structure and data integrity • security barriers • automated design tools • data migration • productivity tools • gateways • SQL dialects • real-time monitoring • data recovery • end-user interfaces

And every month you will have these special columns written by technically competent editors and contributors:

- **Database Design**—tips and techniques for designing complex databases on medium and large systems—both theory and practical implementations of design methodology.
- **Micro Interface**—a detailed evaluation of the new development approaches and products that have migrated up and down between large systems and microcomputers. How to port applications either way.
- **Shoptalk**—offering practical solutions to problems commonly faced by database administrators.
- **For Managers Only**—lively discussions for high level MIS managers on subjects like training, increasing and measuring productivity, and corporate strategies.
- **Product Watch**—objective reviews of new DBMS products from mainframe database managers to development software to productivity tools.
- **Database Library**—reviews of new books and summaries of technical articles from association journals, newsletters and other magazines.

DATABASE PROGRAMMING & DESIGN is guaranteed to help you solve your technical database problems and increase system performance.

Because DATABASE PROGRAMMING & DESIGN is edited strictly for professionals, the press run for the Premier Issue will be very limited. To assure receiving your FREE copy, please return the coupon today.



No-risk charter offer for readers of AI Expert, Computer Language *and* Unix Review

Because you are a valued reader of one or more of our associated technical programming magazines (*AI Expert, Computer Language,* and *Unix Review*) you are entitled to several extra bonuses:

- a FREE Premier Issue.
- a special Charter Subscriber rate, should you decide to continue, of just $37 for a full year's subscription. That's $20 off the regular $57 rate.
- a guarantee that if the Premier Issue does not live up to your expectations you may cancel and owe nothing.

Send to: DATABASE PROGRAMMING & DESIGN, P.O. Box 10716, Des Moines, IA 50309

DATABASE Programming & Design

P.O. Box 10716
Des Moines, Ia. 50309

GUARANTEE: after reviewing my FREE Premier Issue, I may cancel if I do not wish to continue. If I decide to continue I may still cancel at any time and receive a full refund for the issues remaining.

YES! Please put my name on your database as a Charter Subscriber to DATABASE PROGRAMMING & DESIGN, the magazine providing practical, technical solutions to DBMS specialists. I understand I will receive the FREE Premier Issue (while the supply lasts) and if I decide to continue, a full year's subscription for *only $37, a $20 savings* off the basic subscription price.

Note: Canadian delivery, US$43. Other international, US$49—please send payment with order in U.S. funds drawn on a U.S. bank.

Please check: ___ Payment enclosed ___ Send a bill ___ Charge to:

___ Visa ___ MasterCard ___ AmEx Card # _____

Exp. date _____ Signature _____

Name _____

Company _____

Address _____

City _____ State/Prov. _____ Postal Code _____

S7CL

CIRCLE 86 ON READER SERVICE CARD

The *COPY* program has a myriad of useful features. The N option, which allows you to see what an ambiguous file specification would copy without actually performing the copy, is a clever and useful addition. You can copy on the basis of a date and/or time range to include or exclude files in the specified range. A query mode asks you to respond <Y>es, <N>o, <Q>uit, or <ESC> for each file that meets the copy criteria you specify.

The query can be for all files, files that would be overwritten on the destination directory, or files that are new to the destination directory. The program creates the destination directory automatically if it does not already exist. A combination of these options makes it a breeze to do backups from the hard disk.

Another useful option gives you the capability to duplicate the entire directory structure of a disk or directory to your chosen destination.

Running under PC-DOS 2.1, I discovered that Command Plus' *COPY* command would not respond to the standard device names such as *CON:* or *LPT1:*, making it impossible for me to create a quick batch file via *COPY CON: <batchfile>* or send a directory to the printer by *DIR >LPT1:*.

ESP Software Systems hopes to have this problem corrected in its next release. Incidentally, this is only a problem when you use Command Plus with PC-DOS 2.1.

The DOS *COPY* capability of appending files with a plus sign (+) is not supported, but text files can be appended by redirecting the multifile *TYPE* command's output to a file.

DEL in Command Plus has all of the same file selection capabilities as *COPY*. The preview option is extremely useful in the *DEL* command; it can prevent the inadvertent deletion of files.

REN (file rename) renames directories and files and accepts multiple ambiguous file specifications. *MKDIR* (make directory) creates multiple directories with a single command. If you include a command argument for *CLS*, it will clear the screen and set the display attribute you specify.

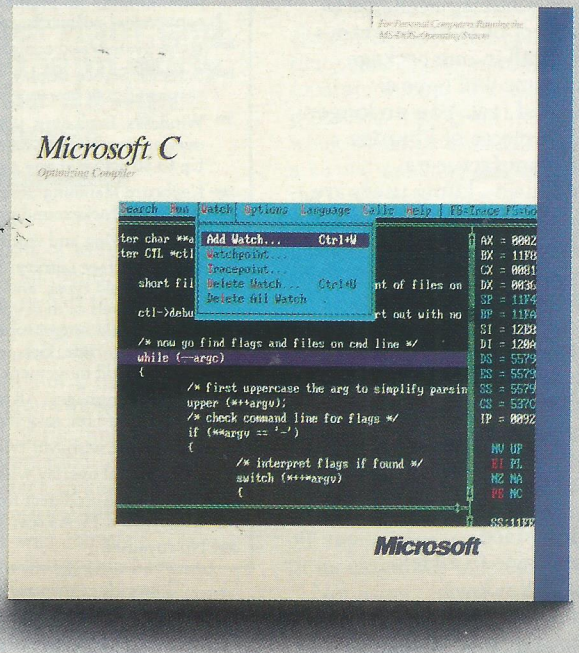
Command Plus provides greatly enhanced command-line editing, allowing you to edit the contents of your command line as you would with a text editor or word processor. Unlike DOS,

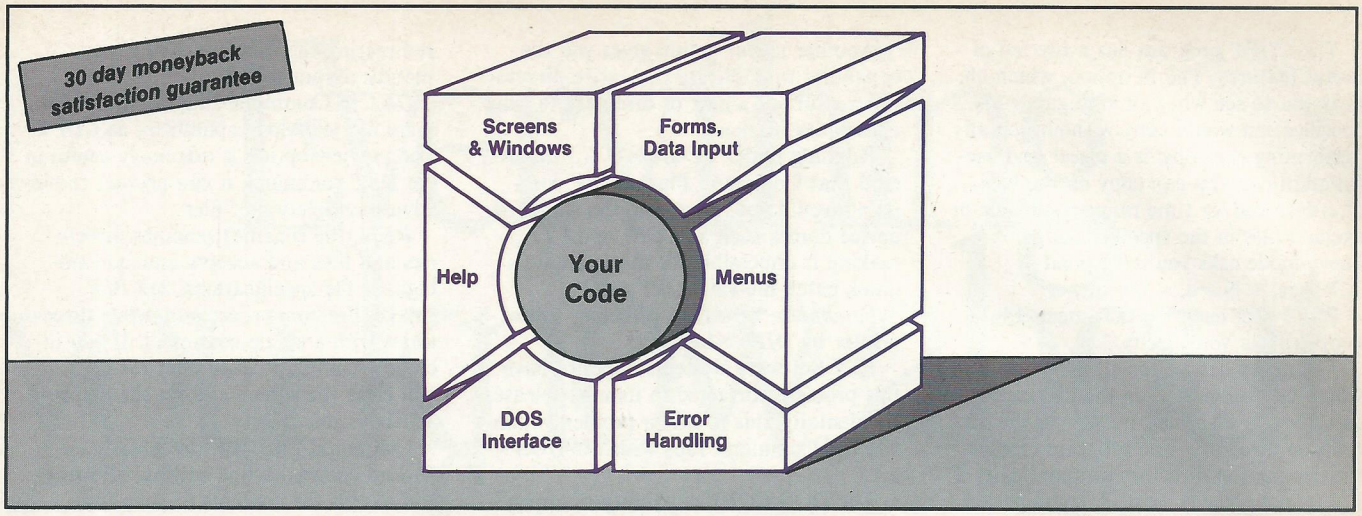
Command Plus commands not in PC-DOS/MS-DOS

ALIAS	Command line macros.
CLRD	Clear directory stack (from PUSHDIR commands).
CLRALIAS	Clear ALIAS assignments.
CLRENV	Clear environment variables.
CLS [attr]	Clear screen [optional attribute].
DATE [spec]	Set/show system date.
DIRS	Display directory stack.
DFS [drive]	Display disk free space.
HISTORY [w | on | off]	Control of command recall feature.
LOG [on | off | entry]	Control of log feature.
MEM	Display amount of memory available.
MOVE	Move file(s) between subdirectories.
POPD	Pop directory from stack.
PUSHD	Push current directory on stack.
TIME [spec]	Set/show system time.
UNALIAS	Remove ALIAS macro.
WHICH <command>	Find executable command.
![! | number | string]	Reexecute command.
!![^E | num ^E | str ^E]	Reedit command.
?command	Show expanded command.
'command	Execute command without alias expansion.

Table 1.

C 5.0
has three features
professional
programmers can't
live without.





C-Worthy Interface Library helps you smoothly pull together all aspects of an excellent Human Interface.

C Programmers: Wrap an Exciting, Bullet-Proof Interface Around Your Code Quickly.

Introducing... C-Worthy® Interface Library

The only human interface package you need. That's what our customers are telling us. One early adopter, Novell, Inc. uses it exclusively in the development of their NetWare® Utilities, which reach over 500,000 users. You see, C-Worthy Interface Library is the only library available to handle every aspect of your program's human interface, all in one package. Now your programs will have a consistent look and feel. You no longer have to integrate pieces of libraries from different manufacturers.

As important as you know users are, you often don't have the time to heavily invest in writing routine code. And that's OK, because with over 400 tight, ready-to-use functions, C-Worthy Interface Library takes care of the tedium and lets you spend your time doing what you enjoy. Concentrate on the heart of your application — features that make it unique, special. Let C-Worthy Interface Library do your:

- Menus
- Error Handling
- DOS Interface
- Context Sensitive Help
- Screens, Windows
- Forms, Data Input (optional)

You control color, size, border, location, etc. And if there's anything you want to change, you can. Source is available to provide you with the flexibility you need. And you can distribute your applications freely, with no royalties.

C-Worthy Interface Library requires hard disk media with 256K RAM. MSDOS 2.0 + and IBM PC, or compatible, TI Professional, NEC APC III, or VICTOR 9000. C-Worthy is a registered trademark of Custom Design Systems, Inc.

Tech Specs

- Compilers: Microsoft 3.0+, Quick, Turbo, Lattice. All models.
- 350+ functions written in C, 75+ in Assembler.
- Menus: Fully support pop-up, Lotus style, MS Windows style (pull-down), pull-up.
- Errors: DOS, program, and user.
- DOS Interface: 62 functions. File handling, dir. and drive management, date & time conversion, wildcards, more.
- Help: System and context sensitive.
- Screens: Screen display, color palettes, save, restore, scroll, more.
- Windows: Exploding, tiled, pop-up, overlapping. Direct video access and virtual. Up to 50 active at any time.
- Keyboard Handling: Regular, function, interrupt, background procedures.
- Editing: String and word wrap text.
- Form Interface Library: 118 functions. Over 15 field types, and user definable field types. 3 levels of data validation: type, multiple field ranges, optional validation procedures. Hide, lock, or secure a field. Optimal field movement.
- Foreign Languages: All text messages in separate files for easy translation.
- Compatible with MS Windows.
- OS/2 special overlay when released.
- Machines: Autodetect for MDA, CGA, EGA, VGA, TI, AT&T, Victor.
- No royalties.

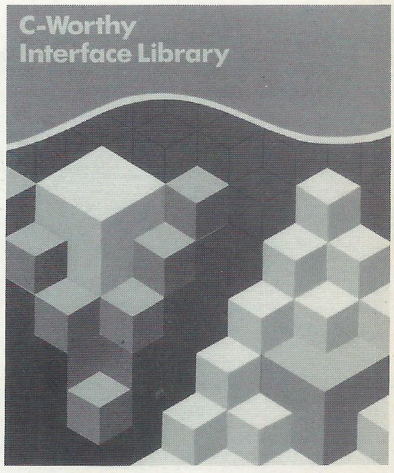
"I heartily recommend this package."
— David A. Schmitt, president, Lattice, Inc.
Over 400 developers in 16 countries already use it.

Thorough Documentation

Indexed alphabetically and by category, the 700+ page Reference Guide includes for each function: an example, description, calling conventions, return values, and related functions. The 250 page User's Guide gets you going with its tutorial and "Getting Started" sections.

CIRCLE 61 ON READER SERVICE CARD

"C-Worthy is a comprehensive C library whose time has come. I heartily recommend it as your next purchase." —Computer Language, 8/87



C-Worthy Interface Library:

Object only	**$ 195**
Form Interface Library add-on	**$ 100**
Object with Forms	**$ 295**
Object with Forms & Library Source	**$ 495**

Please specify compiler and version when ordering.

To Order Call

(800) 821- 2492

in MA (617) 337-6963.

Solution Systems™

541-L Main Street, Suite 410
South Weymouth, MA 02190

Command Plus lets you use the left arrow key to move backward through a command, correct a letter or two, and then not have to retype the rest of the command. Specific keys move the cursor word by word and allow you to go to the beginning or the end of the command line. Similarly, you can delete word by word and to the end of a line.

Command Plus has an undo key and can edit environment variables from the command line. Extended command-line editing is one of the program's best features.

Command Plus supports specification of the drive and path to execute a program, an enhancement if you are running PC-DOS 2.1 or MS-DOS 2.11; DOS 3.0 and later has this feature. Table 1 lists the commands in Command Plus not available in PC-DOS or MS-DOS.

Command Plus supports multiple commands on the command line. Simply enter the commands you want to be executed, separated by the command separator symbol (the default is an ampersand, but you can redefine it to be anything you want), and you have in effect created an ad hoc batch file to accomplish your defined series of commands. The only limitation on this feature is that your multiple commands must fit within the length of the standard command line.

File name pattern matching is much more powerful than that supported by DOS. In addition to the standard asterisk (*) and question mark (?) wildcard symbols, Command Plus allows you to specify specific characters or ranges of characters that are to be included or excluded in file name matching, either in specific positions or throughout the file name. This feature is quite powerful.

Command Plus has a history log that allows you to recall previously used commands to the current command line. Once recalled, you can edit and reissue them. The *HISTORY* command controls this feature. The up and down arrow keys allow you to scroll through the history log. Command recall in conjunction with the enhanced command line editing is a powerful combination.

Another useful command, *LOG,* allows you to record to a disk file all keystrokes as they are entered. The name of the log file is specified by an environment variable. The *LOG* command turns logging on and off. If logging is on, every keystroke from the command line goes to the log file as well as to DOS. If logging is off, you can record entries in the log file with explicit *LOG* commands. Each entry in the *LOG* file contains the date and time of the recorded entry.

The *ALIAS* command allows you to create, edit, and control command-line macros. With *ALIAS* you can define any sequence of keystrokes to any other. For example, *ALIAS root C:* defines a command *ROOT* that is, in effect, sending *C:* to Command Plus's processor. A limitation with *ALIAS* commands, however, is that you cannot pass parameters to them in batch files. This may be remedied in a future version of the program.

Command Plus can insert DOS environment variables in the command line as a macro substitution. In combination with the *ALIAS* capability, this is a powerful tool.

A directory stack is implemented in Command Plus, which lets you save the current default directory for later. This is especially handy for saving your place in a batch file; you don't need explicit knowledge of the directory you're in to return to it at the conclusion of the batch job. *PUSHDIR* and *POPDIR* are internal Command Plus commands. Although programs by these names have been available in the public domain for some time, Command Plus goes one step further by making them resident in the command processor. When you push five entries onto the directory stack, the sixth and subsequent *PUSHDIR* (without an intervening *POPDIR)* will cause the first and subsequent entries to fall into the bit bucket.

In addition to these built-in commands, Command Plus has a new external command, *BROWSE,* which allows

Speed.

Fast Execution Speed.

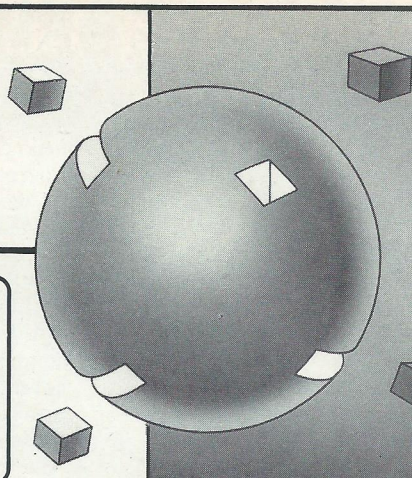
	Microsoft® C 4.0	Microsoft C 5.0
Sieve (25 iterations)	5.7	3.3
Loop	11.0	0.0*
Float	19.9	0.1
Dhrystone	22.8	19.1
Pointer	14.2	7.4

• New optimizations generate the fastest code:
— Inline code generation. NEW!
— Loop optimizations: NEW!
— Loop invariant expression removal. NEW!
— Automatic register allocation of variables. NEW!
— Elimination of common sub expressions.
— Improved constant folding and value propagation.
• Fine tune your programs for even greater speed:
— Coding techniques for writing the fastest possible programs are included in the documentation. NEW!
— Segment Allocation Control:
— Group functions into the same segment to get faster NEAR calls. NEW!
— Specify which segments receive variables to yield faster NEAR references. NEW!
— Uses register variable declarations.
— Mix memory models using NEAR, FAR & HUGE pointers.

*Time is negligible.

Microsoft C 5.0
Optimizing Compiler

C PROGRAMMERS!
THE TOOLS YOU NEED AT A PRICE YOU'LL LIKE



BTree Supports all index file operations. Very quick sequential or random access, duplicate keys, multiple indices, fixed and variable length data records are all supported. **75.00**

ISAM Works on top of BTree to provide a simple, yet powerful application program/file system interface. Complex filesystem manipulation becomes a snap. Provides the power of a database manager with the flexibility of a programming language. **40.00**

NOW MULTI -USER AVAILABLE **60.00**

lp Finally, a completely device independent printer library! lp drives any printer as accurately as possible and allows easy access to its most sophisticated features. Multiple fonts, multi-column output, complex margin formatting, and much more. Pays for itself the first time it's used. **75.00**

Snake The ultimate 'make' utility. We couldn't find a good one, so we wrote a great one. Has all kinds of powerful features including wild card filename expansion, nested macros, and multiple dependency and rules definitions. Ready to go for MS-DOS; C source is there if you use another operating system. **59.00**

Combine & Save: BTree + ISAM + lp **159.00** + Snake **199.00**

Each product includes a typeset manual, example programs, and complete C source code that runs on any operating system. Softfocus products may be incorporated into applications royalty-free.

Credit card orders accepted. Visa, M/C, Amex. Dealer inquiries invited.

softfocus
1343 Stanbury Drive
Oakville, Ontario, Canada
L6L 2J5
(416) 825-0903

CIRCLE 62 ON READER SERVICE CARD

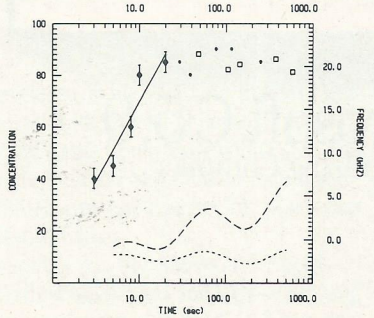
SCIENTIFIC/ENGINEERING GRAPHIC TOOLS
for the IBM PC and compatibles

FORTRAN/Pascal tools: **GRAFMATIC** (screen graphics) and **PLOTMATIC** (pen plotter driver)

These packages provide 2D and 3D plotting capabilities for programmers writing in a variety of FORTRAN/Pascal environments. We support MS, R-M, LAHEY FORTRAN and more. PLOTMATIC supports HP or Houston Instrument plotters. Font module available too!

Don't want to program? Just ask for **OMNIPLOT!** Menu-driven, fully documented integrated scientific graphics. Write or call for complete information and ordering instructions.

GRAFMATIC–PLOTMATIC–OMNIPLOT [S] & [P]



Microcompatibles, 301 Prelude Drive, Silver Spring, MD 20901
(301) 593-0683

CIRCLE 63 ON READER SERVICE CARD

WINDOWS—MENUS—DATA ENTRY—SCREENS

HI-SCREEN *XL*™

If you program, you must try HI-SCREEN *XL* !

HI-SCREEN *XL* is the only tool that gives you all of the following (and much more) whatever programming language you use:

- Unlimited number of screens & windows
- Overlapping windows for menus, data entry, help screens
- Pull-down, pop-up and Lotus-style menus
- On-line help management
- Field checking (for type, range, format,...)
- Field by field and/or full screen input modes
- Scrolling capability
- Full-featured WYSIWYG screen editor
- Redefinable function keys
- Data entry test mode under the editor
- True ease of use, and . . .
- Ease of maintenance: modify screens without recompiling your program.

Includes the HI-SCREEN *XL* Toolbox:
- Capture existing screens from other applications
- Trace compiled programs
- Print screens with clear data field description
- Use screens/windows/menus from DOS and Batch files
- Create titles & logos with predesigned font screens
- ...

Softway, Inc.
PC/Soft Product Line
500 Sutter St., Suite 222
San Francisco, CA 94102

$149 - Risk Free
Visa, M/C welcome
Call (800) 338-2852
in CA (415) 397-4666

For PASCAL, C, dBASE, BASIC, COBOL, FORTRAN

CIRCLE 64 ON READER SERVICE CARD

you to page back and forth in a text file. *BROWSE* is similar to a text editor except you can't make changes to what you see.

BROWSE has a powerful search capability. A search can be accomplished from within the program, or a command-line parameter can take you directly to the first match when the program starts. *BROWSE* does what it does well but overall has less versatility than Vern Buerg's shareware *LIST* utility.

A new command, *WHICH,* is a clever feature I have not seen implemented for MS-DOS before. If you enter *WHICH <progname>* (where *<progname>* is an executable or batch file), Command Plus searches through your current path and reports the full directory location of the file that would be executed.

The Command Plus advanced batch processor is a program called SCRIPT. SCRIPT command files have an extension of S instead of the normal BAT for batch command files. One SCRIPT file can call another and nesting is allowed up to 10 levels. A debug mode tells you what would have been executed by running a SCRIPT command file without actually running it. If you need SCRIPT's capabilities, this program alone is worth the cost of the entire package.

SCRIPT appears to work under the DOS COMMAND.COM command interpreter, too. If you use SCRIPT with COMMAND.COM you must invoke the S files by explicitly executing SCRIPT with the S file as a command line argument (this is done automatically by Command Plus).

SCRIPT is almost a complete programming language, including typed variables (integer, long integer and string), arithmetic, comparison, assignment and Boolean operators, structured programming constructs (*BEGIN/END, CASE, FOR,* and *IF/ELSE),* subroutines, and a library of functions.

If you are a BASIC or FORTRAN programmer, the *GOTO* statement will help you feel at home. The version I tested had a small bug that prevented *GOTO %x* from executing properly.

SCRIPT has just under 50 reserved words, many of which are familiar from other languages. Command Plus is also capable of executing standard .BAT-type batch command files but does this by invoking a secondary copy of COMMAND.COM. SCRIPT's command words are listed in Table 2.

Command Plus occupies a significant amount of disk space and memory. The files needed to run a complete Command Plus occupy about 200K, all of which must be present in the current path. This requirement all but eliminates use of

SCRIPT commands

BASENAME var [var ...]	Strip drive and path from file name.
BEGIN / END	Block structure.
BREAK	Exit WHILE and FOR loops.
CALL label	Call subroutine.
CASE var ... ENDCASE	Case structure.
COMPDATE dt1 dt1 var	Compare dates.
CONTINUE	Transfer control to beginning of loop.
CURSOR line col	Position cursor.
DEC var [var ...]	Decrement variable.
DIRNAME var [var ...]	Extract drive and path from filespec.
DISPLAY char length	Repeats <length> of <char> on screen.
ERRORLEVEL number	Set ERRORLEVEL for exit.
EXECUTE command	Execute command.
EXISTS filename	Test for file in directory/on disk.
EXIT	Exit from SCRIPT file.

Table 2. *(Continued on page 111)*

Speed.

Fast Compilation.
Fast Prototyping.

Microsoft C Version 5.0 includes QuickC,™ which lets you edit, compile, debug, and execute in an integrated environment. It's ideal for prototyping.

- In-memory compilation at 10,000 lines/minute. **NEW!**
- Built-in editor with parentheses, bracket and brace matching.
- Use the integrated debugger to animate through your program, add watch variables and set dynamic breakpoints. **NEW!**
- MAKE file is automatically generated for you. Simply indicate the modules you want to use, then MAKE recompiles and links only those modules that have changed. **NEW!**
- Full C 5.0 compatibility:
  - —Completely source and object code compatible.
  - —Emits CodeView®-supported executables.
  - —Identical compile/link command line switches.

Microsoft C 5.0
Optimizing Compiler

386•386•386•386•386•386•386•386•386•386•386•386•386•386•386•386•386•386•

NOW!

386
C *and* Pascal
for MS-DOS

MetaWare Incorporated announces the *first* available C and Pascal compilers that generate

protected-mode 80386 code

for running on any 80386 machine that runs MS-DOS (e.g., the Compaq Deskpro 386 or the IBM Personal System/2 Model 80). The compilers are functionally identical to our well-respected 8086/286 MS-DOS **High C** ™ and **Professional Pascal** ™ compilers, but now you can get them generating 80386 code.

There's no reason to wait! Industry leaders such as ANSA and Fox Software are already converting their 16-bit database products to 32-bit protected mode, getting increases in speed and functionality. Don't wait years for Microsoft's 386DOS — your competition will have a big jump on you!

Expand your application to the large 32-bit address space and the full 32-bit registers of the 80386. Contact MetaWare for your 80386 software solution today!

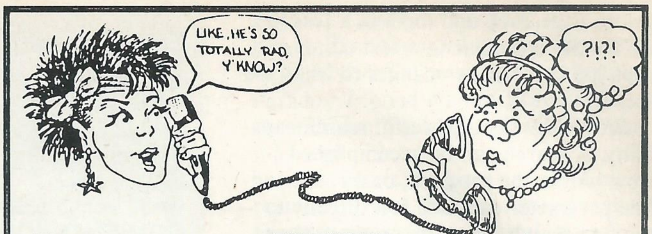
(408) 429-6382, telex 493-0879.

Meta Ware™

INCORPORATED

903 Pacific Avenue, Suite 201 • Santa Cruz, CA 95060-4429

CIRCLE 65 ON READER SERVICE CARD



Now you can use **QPARSER+** to develop compilers, interpreters, complex user-interfaces, language & file format translators (i.e. Pascal to C, Bit Map to Postscript), language debuggers like lint, etc..

Develop language translators in C and Pascal within the IBM PC/XT/AT or VAX/VMS environments. A new user manual, automated syntax tree construction and an advanced code generation language are just a few of the improvements over the original QPARSER.

Another translation by QPARSER+™

Just $475 (PC/XT/AT) — **FREE** demo disk available

QCAD Systems
1164 Hyde Ave., San Jose, CA 95129 (408) 995-6884
Outside Calif. call TOLL-FREE (800) 538-9787

CIRCLE 66 ON READER SERVICE CARD

NEW!
FASTER THAN EVER!
DeSmet C V3.0

FASTER C DEVELOPMENT

Invoke the DeSmet C compiler from the SEE™ full screen editor and the first error will return you immediately to SEE at the error line with the error message displayed.

FASTER COMPILATION

When you don't use inline assembly code or don't want to see the ASM88 output, the V3.0 compiler produces object code directly - making DeSmet C up to ***twice as fast*** as before.

PLUS EXPANDED STANDARD LIBRARY

Networking, path, file, time, enhanced string functions, environment support now included.

FULL FEATURES WITH EVERY PACKAGE
— ONLY $109 —

C Compiler, Assembler, Binder, Librarian, Execution Profiler, Overlays, 8087 and S/W Floating Point, Full STDIO Library and Full Screen Editor (SEE). Debugger and Large Case options available at $50 each.

C Ware Corporation
P.O. Box 428, Paso Robles, CA 93447
Phone: (805) 239-4620 **Telex:** 358185 **BBS:** (805) 239-4834
We accept **VISA, MC & AMEX.** Call now and we'll ship today.
Street Address: 945 Spring #14, Paso Robles, CA 93446

CL 11/87

Modula-2
Database Toolbox

Now with the availability of Solid Software's Modula-2 Database Toolbox, which is based on the most advanced routines known today, you can build extremely FAST database applications quickly and easily.

- B+ tree indexes ■ Memory caching of indexes for fast access
- Manages huge index and data files (up to 4 gigabytes)
- Duplicate keys ■ Fixed and variable length data records
- Fast virtual Quicksort (not limited by available memory)
- Comprehensive database library ■ Many extras (direct screen video module with EGA support, field editor, timer, DOSFiles)

Supports LOGITECH Modula_2 Version 3.0

MakeIndex, OpenIndex, CloseIndex, AddKey, DeleteKey, FindKey, ClearKey, NextKey, PrevKey, MakeFile, OpenFile, CloseFile, AddRec, DeleteRec, GetRec, PutRec, QuickSort, EditString, . . .

PC/MS-DOS 2.0 or higher, 128k RAM
Only $99.95. With source code, $149.95.

Why wait? Order Today!

VISA and MasterCard accepted.

Solid Software, Inc.
9650 US Route 42
Cincinnati, Ohio 45241
(513)779-0009

CIRCLE 67 ON READER SERVICE CARD

Command Plus on a standard PC with 360K floppies. It is possible to run the program on a system with 720K drives, but a hard disk or a large, expanded-memory RAM disk is best.

The cost in terms of RAM is less severe, but not trivial. The COMPLUS command processor, the heart of Command Plus, takes up 43K of RAM (in addition to COMMAND.COM under DOS 2.1). COMPLUS can be run as the default SHELL under DOS 3.0 and later, so you will save the size of a resident COMMAND.COM in that case.

SCRIPT requires 47K of RAM but is not resident when you are not running a SCRIPT batch file. ESP Software Systems explained the reason COMPLUS cannot be used in lieu of COMMAND-.COM under DOS 2.1: The code to execute the DOS *EXEC* function (4Bh) is actually in COMMAND.COM in DOS 2.1 but was moved to the DOS code beginning with version 3.0.

Command Plus has a few quirks, but some of them are excusable when you appreciate the difficulties DOS throws in the path of the ambitious programmer. My testing was conducted using DOS 2.1. Command Plus seems compatible with most of the memory-resident software I tried out but did not recognize the existence of a path enhancer and processed a *PROMPT* string differently than DOS does.

One shortcoming of the current version is the lack of a way to lock COMPLUS into the system when it is invoked using DOS 2.1. You can exit from COMPLUS when it is running as a secondary command processor (always true under DOS 2.1) by issuing an *EXIT* command from the DOS prompt. The program should be locked in place to prevent you from shooting yourself in the foot if you issue an *EXIT* command after invoking a terminate-but-stay resident program while COMPLUS is loaded. ESP Software Systems says this drawback should be corrected in a forthcoming update.

I ran my memory-resident programs under COMPLUS, not above it, which worked fine except that some TSRs that rely on being the last to use a specific interrupt might not work correctly if COMPLUS has put itself between the resident program and the interrupt vector. I did not experience this problem, however.

The Command Plus documentation looks good. The manual consists of about 80 pages of typeset material with a table of contents and an index. I found the manual to be disorganized when learning to use the programs but adequately organized as a reference. More examples illustrating the use of each command would improve the user's understanding,

EXT var [var ...]	Extract extension from file spec.
FDATE fname var	Return date of fname file.
FOR var [WITH attrib] IN(set)	Loop structure.
FSIZE filename var	Return time of filename file.
FTIME filename var	Return time of filename file.
GETENV varname var	Return current value of environment var.
GOTO label	Immediate command transfer to label.
IF(conditional)...ELSE	Conditional structure.
INC var [var ...]	Increment variable.
INENV varname	Test if varname is in Environment.
INSTR source strpos substrng pos	Test for or find pos of substring.
ISARCH filename	Test archive bit set for filename.
ISDIR filename	Test if filename is a subdirectory.
ISHID filename	Test if filename is a hidden file.
ISSYS filename	Test if filename is a system file.

Table 2. *(Continued from page 109)*

And speed.

Fast Debugging.

Microsoft C Version 5.0 includes Microsoft CodeView, our source-level windowing debugger that lets you debug more quickly and thoroughly than ever before.

- Debug larger programs:
  - —Debug through overlays created by the Microsoft overlay linker. NEW!
  - —Expanded Memory Specification (EMS) support. NEW!
- Fast debugging through precise control of your program execution:
  - —Access source level and symbolic debug information from your Microsoft C, FORTRAN, and Macro Assembler programs. NEW!
  - —View your source code and assembly simultaneously.
  - —Watch the value of variables change as you execute.
  - —Set conditional breakpoints.
  - —Animate or single step through your program.
- CodeView brings you as close as you've ever been to your hardware:
  - —Swap between your code and output screens.
  - —Watch your registers and flags change as your program executes.

All benchmarks run on an IBM® Personal System/2.™

NOW SHIPPING

Microsoft®

For your free C 5.0 information packet, call:

(800) 426-9400.

In Washington State and Alaska, (206) 882-8088. In Canada (416) 673-7638. Microsoft, the Microsoft logo and CodeView are registered trademarks and QuickC is a trademark of Microsoft Corporation. IBM is a registered trademark and Personal System/2 is a trademark of International Business Machines Corporation.

CIRCLE 68 ON READER SERVICE CARD

Throw Away Your Flowcharts!! Cut Development Time By Up To 30%

The GeneratoR is the online design tool, based on the popular Warnier/Orr diagrams that can generate source code and program documentation from completed design diagrams.

GeneratoR
- Generates
  - Design Diagrams
  - Source Code
  - Documentation
- Languages
  - C
  - Pascal

Introductory Special

Pascal GeneratoR :$90.95 C GeneratoR :$104.95

C & Pascal GeneratoR :$149.95 Demo GeneratoR :$10.00

C or Pascal Source Code Convertor :$39.95
Diagram Editor: $39.95

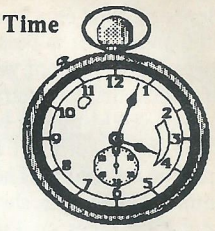
FREE -Source Code Converter and Diagram Editor with GeneratoR Purchase

For all orders, add $1.50 Shipping and Handling. Cash, Check, COD, Visa and MC accepted. In Kansas add 5% sales tax

Call SoftWare Support 913-354-1618
701 Jackson B-5 Topeka, Kansas 66603

CIRCLE 69 ON READER SERVICE CARD

Along With Your Computer, Your Time is the Most Important Thing You Own. . .So Why Waste It?



Quilt Programmer Productivity Tools will help you manage your software projects and get control of your time!

SRMS™
Software Revision Management System

- Full Featured Revision Control System
- All Versions stored in a Single ASCII File
- Support for Unlimited Libraries
- Support for all programming languages
- Allows you to use your current compilers and editors without conflict
- Windowing Shell interface to simplify use
- MERGE facility to consolidate different development paths easily, while pointing out conflicting areas
- Full audit trail tracking and reporting on all library components
- Handles big programming projects easily
- Full DOS Pathname and Environment Variable Support
- Requires DOS 2.1+, 224 K, F/H Disk

SRMS Version 3.0.....$185

QMAKE™
Intelligent Program Generation Utility

- Controls the rebuilding of even the most complex systems
- Relieves the developer of remembering which modules need to be rebuilt based on recent changes, how to rebuild them, and in what order to rebuild them
- Works with most compilers, assemblers, and linkers
- Supports full macro definitions, UNIX make-file compatability, recursive invocations, and command line parameters
- Interfaces completely with SRMS, providing you with a complete set of productivity tools to handle any size project
- Requires DOS 2.1+, 128K F/H Disk

QMAKE Version 1.2.....$99

SRMS + QMAKE $250

NEW ! TXT Tools NEW !

QSE - Quilt Text Stream Editor
QSRCH - Quilt File Search Utility (Like UNIX GREP)
QDIFF - Quilt Windowing File Difference Utility

TXTTOOLS Version 1.0..... $85

QUILT COMPUTING
7048 Stratford Road
Woodbury, MN 55125
(612) 739-4650

Volume Discounts and Dealer Inquiries Welcome

CIRCLE 71 ON READER SERVICE CARD

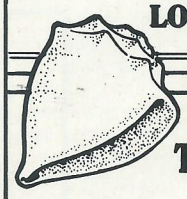
KYOTO COMMON LISP and GNU EMACS have been merged into one high-performance LISP & C programming tool!

FOR SUN CARTRIDGE TAPE OR MICROPORT SYSTEM V FLOPPIES INCLUDING ALL SOURCE CODE
SEND $250
free shipping on prepaid orders
MASTER CARD/VISA

THE AUSTIN CODE WORKS
11100 LEAFWOOD LANE
AUSTIN, TEXAS 78750-3409
512-258-0785

CIRCLE 70 ON READER SERVICE CARD

Users can add new functions to LOTUS, WORDSTAR, and dBASE



Move into
THE WEINER SHELL™
Get Room to Grow

- Users can add their own functions to C, BASIC, PASCAL, LOTUS, WORDSTAR, dBASE & most other programs on the market.
- Custom-design memory-resident windows, menus, screens & utilities.
- Easily set up your own context-sensitive help screens and light-bar menus.
- Run Shell programs without leaving your application.
- Execute Shell programs automatically, at timed intervals, or with interrupt-driven serial conditions.
- 60K memory required. Supports up to 8M bytes of Lotus-Intel memory.
- Customization available.
- $199 (includes limited no-royalty agreement).

THE FIRST MEMORY-RESIDENT PROGRAMMING LANGUAGE

GRYPHON®
MICROPRODUCTS

P.O. Box 10087/Silver Spring, MD 20904/(301) 384-6868

CIRCLE 72 ON READER SERVICE CARD

especially during the learning phase. You should read the entire manual at least once to have a good grasp of the programs.

The manual is not clearly written in many places and suffers from some gross omissions. For example, the *COPY* command is not shown as having the capability to copy to a device or a differently named file as the destination, even though these capabilities do exist (except for the DOS 2.1 bug described earlier). An example of the unclear style appears on page 57: "Following a GOTO, SCRIPT assumes a cleared state and will not necessarily allow some of the constructs encountered within a loop or block of commands." There is no further elaboration on this statement.

One other aspect of the documentation I found annoying was that I received v. 1.1 of the program with v. 1.0 manual plus addenda pages. Furthermore, the addenda pages were 8 1/2 x 11-inch pages while the manual's pages were computer paper size, so I had to run the addenda pages through a reducing copier before I could put them in the manual. The changes from v. 1.0 to v. 1.1 were not trivial, which made learning to use Command Plus much more difficult.

ESP Software Systems reliably returned my calls, and I am impressed that these folks are genuinely interested in helping the user and improving their product by paying attention to bug reports and suggestions for improvement.

Command Plus is a reasonably solid product that should improve with time. Whether or not you need Command Plus depends on what system enhancements you are using. If, for example, you already have Chris Dunford's CED (free) or PCED (shareware) for command-line enhancement; Seaware's EBL (shareware) for improved batch language processing; or Vern Buerg's LIST (shareware), DOS 3.2 (for *XCOPY*), or any of several public domain SWEEP-type programs for enhanced *DEL* functions, then you might not need Command Plus.

If, on the other hand, you want these capabilities and don't want the hassle of ordering or downloading several different programs, then Command Plus could be for you. Assuming you already have DOS, the cost of acquiring the lowest priced combination of the programs I mentioned (CED, EBL, and LIST) is only a few dollars less than the list price of Command Plus.

In any case, Command Plus is a programmer's and hacker's tool. If you need to provide your end user with a system that has a simple user interface, one of the many DOS command shells or menu packages is a better means of implementing that goal.

By Jim Kronman

NAME var [var ...]	Extract the name portion of file spec.
PAUSE [message]	Wait for keypress.
READ var	Get string input from STDIN.
RETURN	Transfer back from subroutine.
SHIFT	Next 10 variables; same as BATCH in DOS.
STRLEN stringvar intvar	Return length of stringvar.
SUBSTR source dest startpos endpos	Extract substring.
TEMPFILE var [var ...]	Create temporary file name.
TOLOWER var [var ...]	Convert to lower case.
TOUPPER var [var ...]	Convert to upper case
VERBOSE on | off	For debugging: write each command to STDOUT.
WHILE(conditional)	Loop structure.
WRITE [@filename] string	Send string to STDOUT or file.

Table 2. *(Continued from page 111)*

And here's the speediest way to get C5.0:

Just visit any Egghead store. There's more than 70 nationwide. And each one has C5.0 in stock, at a discount price.

Or, just give us a call on our toll-free number. We'll send you C5.0 by mail. Within 72 hours.

It's the fastest way we know to get a great deal on a very fast compiler.

Call 1-800-EGGHEAD

Discount EGGHEAD SOFTWARE SM

NGS FORTH

A FAST FORTH, OPTIMIZED FOR THE IBM PERSONAL COMPUTER AND MS-DOS COMPATIBLES.

STANDARD FEATURES INCLUDE:

- 79 STANDARD
- DIRECT I/O ACCESS
- FULL ACCESS TO MS-DOS FILES AND FUNCTIONS
- ENVIRONMENT SAVE & LOAD
- MULTI-SEGMENTED FOR LARGE APPLICATIONS
- EXTENDED ADDRESSING
- MEMORY ALLOCATION CONFIGURABLE ON-LINE
- AUTO LOAD SCREEN BOOT
- LINE & SCREEN EDITORS
- DECOMPILER AND DEBUGGING AIDS
- 8088 ASSEMBLER
- GRAPHICS & SOUND
- NGS ENHANCEMENTS
- DETAILED MANUAL
- INEXPENSIVE UPGRADES
- NGS USER NEWSLETTER

A COMPLETE FORTH DEVELOPMENT SYSTEM.

PRICES START AT $70

NEW◆HP-150 & HP-110 VERSIONS AVAILABLE

NEXT GENERATION SYSTEMS
P.O.BOX 2987
SANTA CLARA, CA. 95055
(408) 241-5909

CIRCLE 73 ON READER SERVICE CARD

Bas_C

System requirements: PC-DOS or MS-DOS 2.0 or later, 256K RAM, two disk drives

Price: $375—commercial model, $199—economy model, $35—demo model (free demo disk available), $59—general manager (Bas_C library)

Available from: Gotoless Conversion, P.O. Box 835910, Richardson, Texas 75083, (214) 404-1404

Support: Telephone and mail support, 30-day money-back guarantee, not copy protected

Bas_C is a BASIC-to-C program translator. Although it could be used for learning C, it is geared more to a business or programming shop switching from BASIC to C.

What are some of the reasons to change languages? Speed is a prime factor. In a comparison, interpreted BASIC code took 3 minutes and 15 seconds, while the compiled C code executed in 2 seconds. Other factors include third-party support in C and easier maintenance of code because of modular structured programming. Whatever your reasons for switching, Bas_C and its related libraries will take most of the drudgery out of the changeover.

A fair knowledge of BASIC is necessary to get the maximum performance from Bas_C. Some familiarity with C will also be helpful, although Gotoless's C libraries provide much of the information you'll need.

These libraries, called managers, duplicate most of the BASIC functions and features not normally found in C. A general library and a graphics library are available now, and communications and machine-language libraries are being developed. The general manager covers about 90% of BASIC's features not found in C and is required to use the other libraries. The libraries support Lattice, Manx, Eco-C88, and Instant C. Without the managers you would have to code a library to duplicate BASIC calls. The time and trouble this would take more than justifies the purchase of Gotoless's libraries.

Bas_C comes on four disks that include Compute's book *From BASIC to C*

Sample of dead code listing

```
***** Dead Code Listing *****

 Line#.Mult   Statement Name
    420. 1     IF
    420. 2     PRINT
    420. 3     RESUME
    430. 1     IF
    430. 2     LOCATE
    430. 3     PRINT
    440. 1     RESUME
```

Figure 1.

Final report of sample program using Bas_C

Statistics: label	Source Program	Target Program
Compiler	IBM BASICA (tm)	Kernighan/Ritchie C
Number of GOTO	6	3
Number of LABEL	4	3
Number of IF	6	3
Number of DO-WHILE	Ø	1
Number of WHILE	Ø	Ø
Number of FUNCTION	1	2
Number of DEADCODE	7	Ø
Number of LINE	38	127

Figure 2.

Oasys presents

Microsoft® cross C

for VAX, Sun, Apollo . . .

VAX SUN APOLLO

OASYS is proud to announce the immediate availability of the OASYS/Microsoft Cross C Development System. Microsoft C, MASM (Assembler), and LINK (Linker) now run on DEC VAX (VMS and Ultrix), Sun and Apollo systems.

Those accustomed to using these superior Microsoft tools on a PC can now build MS-DOS applications on a VAX or workstation. OASYS guarantees that the unsurpassed speed, compactness, and flexibility of Microsoft C have been preserved. The OASYS/Microsoft Cross C Development System offers identical functionality to Microsoft C -- no short-cuts, no alterations -- repackaged to meet today's demands for high performance/low cost development on non-MS-DOS systems.

With the OASYS/Microsoft Cross C Development System you can maintain, or even extend, applications originally created on a PC. Software development teams can now build large, complex MS-DOS (soon OS/2) applications on powerful centralized VAXs or networked workstations.

Regardless of where you choose to do development, OASYS provides the best tools, on the widest variety of hosts, with comprehensive support. Our exclusive relationship with Microsoft, the world's leading supplier of MS-DOS software products, is evidence of our commitment to provide evolving PC tools to OASYS customers.

Prices start at $1,000. New ports are underway. Call today for more information. OEM and end-user inquiries are encouraged.

CIRCLE 74 ON READER SERVICE CARD

Oasys

Microsoft®

230 Second Avenue, Waltham, MA 02154 **(617) 890-7889**

MS-DOS, Microsoft and the Microsoft logo are registered trademarks of Microsoft Corp. Apollo is a trademark of Apollo Computer Inc. Trademarks are also acknowledged to DEC, Sun Microsystems, Inc., XEL, Inc.

FINALLY!

EASYFLOW

An on-screen flowchart processor that knows about flowcharts - not just another "screen draw" program that makes you do most of the work. EASYFLOW is a powerful full-screen graphics program dedicated to flowcharts and organization charts. With it you can quickly compose charts. More important, you can easily modify charts so they are always up to date.

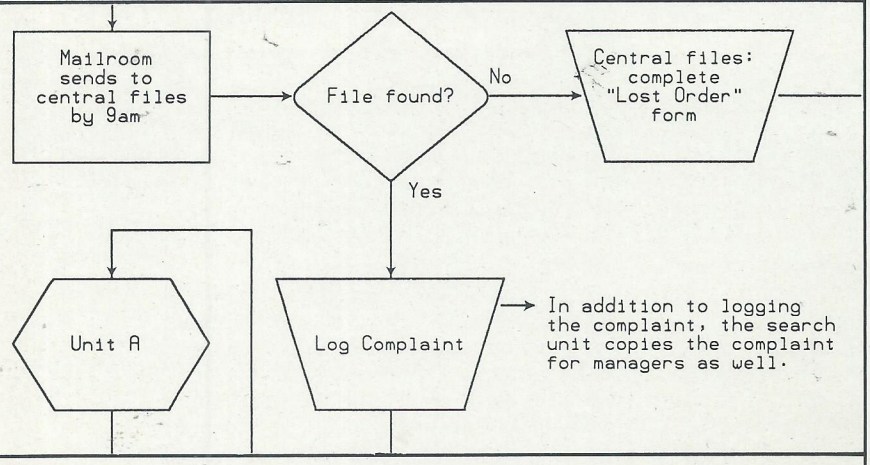
- **Automatic:** Fully automatic text centering within shapes, both horizontally and vertically. Fully automatic line routing & re-routing.
- **Fast:** Written in assembly language for speed.
- **Large:** Charts up to 417 columns wide by 225 lines high. Chart too large for your printer? *EASYFLOW* automatically breaks the chart up & prints it in page size pieces.
- **Standard:** All standard flowcharting shapes included. Custom shapes can be ordered.
- **User friendly:** Don't take our word for it. PC Magazine* says "*EASYFLOW* lives up to its name. It's hard to imagine any easier and more flexible way to produce basic and even complex flowcharts".
- **Mouse:** Optional but fully supported.
- **It prints:** On most popular matrix printers including IBM, Epson, Toshiba, HP LaserJet, LaserJet-Plus and many others.
- **It plots:** On HP7440, 7475, 7550, 7570, 7585B and compatible plotters.
- **It works:** We are contractually prevented from mentioning the name of the "big eight" accounting firm that purchased a world-wide site license, but we can tell you that they spent months evaluating all available flowcharting packages before choosing *EASYFLOW*.
- **Rush delivery:** Order by noon today (eastern time) and we'll have it to you by courier tomorrow**. Rush delivery charge is $15.00 (instead of $2.00) and is available only in USA & Canada.
- **Documented:** 100 page manual plus over 150 screens of context sensitive help.

EASYFLOW works on IBM PC's, IBM PS/2 and compatibles. Requires 384 K memory, DOS 2.0 or higher and an IBM CGA/EGA/VGA or Hercules monochrome compatible adapter card.

Order direct for only **$149.95** + $2.00 S&H (USA/Canada), $10.00 (foreign). Payment by M.O., cheque, VISA, Mastercard or Company PO.

* March 10, 1987 issue, page 278.

** Rush orders are shipped by Purolator Courier and normally arrive the next business day to most locations. Remote destinations take longer.



The chart fragment above was produced on an HP LaserJet-Plus and is actual size and unretouched. Publication quality charts like this can be produced using only minutes of preparation time and seconds of print time.

HavenTree Software Limited
PO Box 1093-K
Thousand Island Park, NY 13692

Order Desk: 1-800-267-0668
Information: (613) 544-6035 ext 46
Telefax(G3): (613) 544-9632

CIRCLE 75 ON READER SERVICE CARD

and a demo disk. Although features were added to the previous release (v. 2.2), the code size in this version was cut in half.

Bas_C operation is batch-file driven. Source, target language, and target code layout are selected from a menu. Source selection allows IBM's BASICA and ANSI BASIC. All source must be in ASCII code, so any programs you want to convert must be saved with BASICA's A switch or entered with an appropriate editor. Bas_C supports target languages from Lattice, Manx, and ANSI (Kernighan and Ritchie). The documentation mentions support for Eco-C88 and Instant C, but they are not included on the menu. There is also no mention of which selection would support Eco-C88 or Instant C.

Since straight Kernighan and Ritchie code is also supported with Bas_C, most C compilers should be able to handle the generated code. You will have to recode the routines in the general manager yourself if your favorite compiler is not currently supported. Gotoless plans to support other C compilers, including Microsoft C 4.0, in a future release. Gotoless also plans to add a source language extender to Bas_C to help translate other BASIC dialects.

The documentation is the weakest part of this package since some needed information is in disk files instead of in the 15-page manual and 21-page overview.

Bas_C converts a program in several phases. The first phase checks the source program's syntax and flags any errors in your BASIC source, including assignments that are legal in BASIC but not in C. Errors the BASIC interpreter might miss are caught. The remaining passes analyze program flow, eliminate dead code, and convert the BASIC program into a modular, structured form in the target language. Variables are scoped according to how the source program accesses them. If a variable is accessed by more than one *goto* or *gosub*, it is declared global. The remaining variables are declared local to the subroutines in which they appear.

Messages during the operation of Bas_C are very informative and cover syntax errors in the BASIC source, type mismatches in the target, dead code removed from source, and a final report on conversion. All error messages, warnings, and reports are output to the screen but may be sent to the printer or a disk file by I/O redirection. This option is mentioned in a text file on the demo disk, which includes examples and a list of which files normally output to the screen. Figure 1 is a sample of the dead code listing and Figure 2 is the final report on one of the programs I converted.

The programs I used to test Bas_C were a simple labeler used by the Chica-

go Computer Society for public domain library disks, the infamous Sieve of Eratosthenes, and a public domain maze game. I tried all three target versions of C with the default selection of Lattice C first since I have a Lattice compatible C compiler (Optimum-C 3.1 from Datalight). I also used Microsoft C 4.0.

One minor problem I had was in the translation of the BASIC *LPRINT* statement. For example:

```
190 LPRINT CHR$(27) ; "G" ; A$
   200 LPRINT
```

translated to:

```
lprintf("%s%s%s\n", chr_str(27), "G",
   a_str) ;
lprintf("\n") ;
```

This mistranslation occurred with all three target selections. The translated code is not a legal C statement and is not defined by the ANSI standard, Kernighan and Ritchie, or in Gotoless's general manager. *LPRINT* should translate into one of the following:

```
FILE *pr ;
pr = fopen("PRN:") ;
fprintf(pr,STRING) ;
```

or

```
fprintf(stdprn,STRING) ;
```

Gotoless is working on this translation problem and will try to have a solution by the next release of Bas_C.

I should note that *ON ERROR* statements are commented out and related *error* statements are removed as dead code. For example:

```
310 ON ERROR GOTO 420
420 IF ERR=27 THEN PRINT "TURN
   PRINTER ON":RESUME
430 IF ERR = 68 THEN LOCATE
   25,1:PRINT "PRINTER NEEDS ATTN"
```

In the previous example, line 310 was commented out and lines 420 and 430 were removed as dead code (Figure 1). Thus, any error checking set up in your BASIC program must be ported over by hand.

The only other conversion problem I had occurred with a Sieve of Eratosthenes program. The resulting C program compiled but crashed with a run-time memory error. Because the BASIC code's *DIM* statement was referenced only once, it was translated as local to the function instead of globally as needed. This minor glitch resulted from a problem in the C language itself and not with Bas_C. The error is easily corrected by declaring flags global to the program. One solution is to add dummy calls to your BASIC program to cause a proper translation where needed, as demonstrated on the demo disk. Another option would be to hand-modify the C code.

Two other points to consider are BASIC's automatic dimensioning of arrays of fewer than 11 elements and automatic initializing of variables to 0. C does not normally initialize declared variables, with the exception of static variables. Bas_C requires all source code arrays to be explicitly declared with the *DIM* statement.

Conversion with Bas_C was surprisingly fast. Even the conversion of the maze program, which was several hundred lines containing close to 2,000 BASIC statements, took less then eight minutes. The resulting C program compiled without a hitch. In fact, most of my troubles with the maze program were in the linking stage. It seems that my C compiler is not quite as Lattice-compatible as I thought. Once I finish adding the missing functions to the general manager, I'll be able to convert just about any BASIC program.

Overall, the quality and support of Bas_C are outstanding. Gotoless answered my many questions and even sent me an upgrade that incorporated many of my suggestions. Since the upgrade contained other improvements as well, it is clear that Gotoless listens to its customers. Most of the glitches I found in the original version (1.1) I had received to review have since been removed.

Unless you need to covert many large programs, I would recommend the economy model because of its low price. A programming shop is more likely to need and be able to afford the commercial model. A 100-line conversion sample is available from Gotoless for $1 when you send the company your sample BASIC code. Call or write for further details.

Update: Versions 2.4 and 2.5 of Bas_C support Microsoft C and Turbo C. Bas_C v. 2.5 also supports Lattice C, Instant C, and Aztec C, and includes 200 pages of documentation.

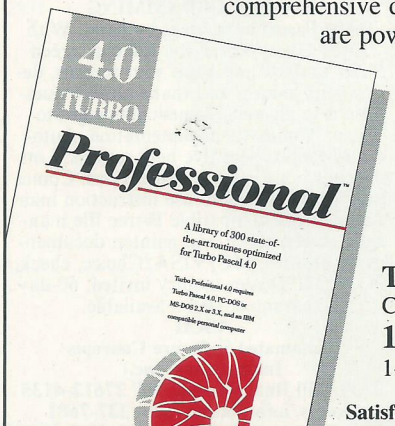
By Glenn J. Wiorek

New!

Hire a Pro for Your New Turbo 4.0

Turn on the power of Turbo PROFESSIONAL 4.0, a library of more than 300 state-of-the-art routines optimized for Turbo Pascal 4.0. You'll have professional quality programs finished faster and easier.

Turbo PROFESSIONAL 4.0 includes complete source code, comprehensive documentation and demo programs that are powerful and useful. The routines include:



- Pop-up resident routines
- BCD arithmetic
- Virtual windows and menus
- EMS and extended memory access
- Long strings, large arrays, macros, and much more.

Turbo PROFESSIONAL is only $99.
Call toll free for credit card orders.
1-800-538-8157 extension 830
1-800-672-3470 extension 830 in CA

Satisfaction Guaranteed or your money back within 30 days.

Turbo Pascal 4.0 is required. Registered owners of Turbo Professional by Sunny Hill Software may upgrade for $30. Include your serial number.

For other information call 408-438-8608, 9 AM to 5 PM PST. Shipping & taxes prepaid for US and Canadian customers, others please add $6 per item.

TURBO Power

TurboPower Software 3109 Scotts Valley Dr., Suite 122 Scotts Valley, CA 95066

CIRCLE 76 ON READER SERVICE CARD

CLASSIFIED CONNECTION

USER INTERFACE PROTOTYPING,
construction, run-time management tool for PC applications. Systems builders report 50% savings in interface development using SKYLIGHTS compared to programs like Microsoft's Windows development kit. For C programs; no coding; unlimited look and feels: character-based, monochrome displays; data entry screens; iconic/textual menus; overlapping windows; full color; high-resolution graphics; lots of pointing devices. Upcoming release to run under WINDOWS. Versions for all IBMs and compatibles, now including PS/2 with VGA boards. Versions from $195. No run-time fees.
The Report Store
910 Massachusetts St., Suite 602
Lawrence, KS 66044
(913) 842-7348

CIRCLE 86 ON READER SERVICE CARD

LANBASE TOOLBOX FOR
Turbo Programmers! Share data base written in Turbo Pascal over a LAN. Turbo v. 3.0, DOS v. 3.1, and a LAN that supports file sharing (DOS SHARE command) required. New LANbase ToolBOX PLUS is Database Toolbox-compatible. LANbase ToolBOX is $99.95, LANbase ToolBOX PLUS is $249.95. VISA/MC accepted.
NIGHT OWL COMPUTER SERVICES
Software and Consulting
P.O. Box 138037, Toledo, OH 43613
(419) 474-9248

CIRCLE 87 ON READER SERVICE CARD

MULTITASKING/PROCESSING
TURBO PACKAGE for Turbo Pascal allows 640K progs/computer. Any MIMDA setup. Network with a modem or RS232/422. Easy to use. 90-day money-back guarantee. TP-640K $59.95, TP-640K-TASK $89.95, TP-640K-TASK-NETWORK $189.95. +tax+shipping (USA $5, Foreign $10). VISA, M/C, or COD.
Conversational Computer Systems
5371 Verbena Rd., San Antonio, TX 78240
(512) 692-0353

CIRCLE 88 ON READER SERVICE CARD

35MM SLIDE FROM YOUR PC
COMPUTER SLIDE EXPRESS converts graphic files produced on the IBM PC into brilliant 35mm color slides with color resolution 400% better than your monitor. Leave your printouts behind. Use high-resolution color slides up to 4000-line. COMPUTER SLIDE EXPRESS $9/slide.
VISUAL HORIZONS
180 Metro Park
Rochester, NY 14623
(716) 424-5300

CIRCLE 107 ON READER SERVICE CARD

DEVELOPERS, NOW AVAILABLE!
Sof-Tel Inc., in their goal to provide innovative Software Solutions, now has ready for you two new IBM PC software packages. Do you need Modem dialing and interrupt Comm port support? Get AUTODIAL! User-definable script files allow use with many modems. Send $49.95 plus $3 P&H. Also available, STAGE2-PC, the PC version of the mainframe macro processor STAGE2. Disk plus printed manual only $49.95 plus $3 P&H. (Florida residents add 5% sales tax.)
Sof-Tel Inc.
P.O. Box 5116, Lighthouse Pt., FL 33074
(305) 942-6671

CIRCLE 89 ON READER SERVICE CARD

INCREDIBLE ASSEMBLER BARGAIN!
Turn your PC into a development system with this one-time offer of five sophisticated cross assemblers for less than the price of one! For just U.S. $150, you will get:
Z80ASM—for Z80 microprocessor family
85XASM—for 8080/8085 microprocessors
96XASM—for MCS-96 microcontroller family
51XASM—for MCS-51 microcontroller family
48XASM—for MCS-48 microcontroller family
Send for more information today!
LEAR COM COMPANY
2440 Kipling St., Suite 206
Lakewood, CO 80215
(303) 232-2226

CIRCLE 90 ON READER SERVICE CARD

MINIMIZE TURBO PASCAL DEBUG TIME
Tmark allows Turbo to continue compiling after an error is found without returning to line one. Tmark can save and restore Turbo's state during a compilation. As each compiler error is found you can fix it and continue compiling. Saves are made automatically before compiler errors or at lines designated with a {tmark} comment. Tmark dramatically reduces debug time. Once you try it you will never want to give it up! $80+$2 s/h, VISA/MC.
TANGENT DESIGNS
P.O. Box 896, Lake Forest, IL 60045
(800) 356-2750, (312) 295-0030

CIRCLE 91 ON READER SERVICE CARD

ROMABLE CODE ON PC!
PCLOCATE allows PC users to develop ROM-based software from MS-DOS EXE files. The user specifies the physical location of all segments. Output files are compatible with most PROM programmers. PCLOCATE supports the 8086, 8088, 80186, 80188, and 80286 processors. MC/VISA.
ALDIA SYSTEMS INC.
P.O. Box 37634, Phoenix, AZ 85069
(602) 866-1786

CIRCLE 92 ON READER SERVICE CARD

STOP PROGRAMMING
Turbo Pascal data bases by hand. With Turbo GhostWriter you use our screen editor to draw and paint your screens, define your indexes and that's all! No questions to answer. No new languages to learn. Automatic documentation. Automatic context-sensitive help screens. Lots of hooks built in for modifications. Comes complete with easy-to-use instruction manual, network compatible B-tree file manager, screen editor and painter, documentation utilities. MC/VISA/Choice, check, PO, COD. Dealer inquiry invited. 60-day guarantee. Demo available.
ASCII
(Automated Software Concepts International Inc.)
3239 Mill Run, Raleigh, NC 27612-4135
Orders/information (800) 227-7681

CIRCLE 93 ON READER SERVICE CARD

EDITINGTOOLS 2.1
Now you can have a slick full-screen multifile text editor for only $35. Enjoy the ease of editing many files at once and the freedom of moving text among files. File sizes are limited only by memory. With an easily reconfigurable keyboard and a convenient multidirectory DOS shell. Optimized Turbo Pascal source code is available for $134 more. Add $4 for s/h.
Jou Laboratories
P.O. Box 5059, San Luis Obispo, CA 93403

CIRCLE 94 ON READER SERVICE CARD

INLINETOOLS 2.0
Turbo Pascal inline programmers! Now you can write at ease DOS/BIOS calls, critical statements, and procedures in tight and fast assembly code right into your Turbo Pascal program text. No more DEBUG and MASM. Just write assembly code as comments, and let IT append inline machine code for you instantly. $65+$4 s/h.
Jou Laboratories
P.O. Box 5059, San Luis Obispo, CA 93403

CIRCLE 95 ON READER SERVICE CARD

CROSS ASSEMBLERS
Relocatable, PC-Compatible, Conditional Assembly Macros, Fast, Reliable, Built-in editor. For most microprocessors...from $150. Also: Cross Debug/Simulators, Eprom Programmer Board
MicroComputer Tools Co.
912 Hastings Dr., Concord, CA 94518
(800) 443-0779
In CA (415) 825-4200

CIRCLE 96 ON READER SERVICE CARD

OPT-TECH SORT/MERGE
Extremely fast Sort/Merge/Select utility. Run as an MS-DOS command or CALL as a subroutine. Supports most languages and filetypes including Btrieve and dBASE. Unlimited file sizes, multiple keys, and much more! MS-DOS $149. XENIX $249.
Opt-Tech Data Processing
P.O. Box 678, Zephyr Cove, NV 89448
(702) 588-3737

CIRCLE 97 ON READER SERVICE CARD

ROM 8086 CODE
ROM KIT locates EXE modules developed on a PC. ROM KIT is ROMable so 8086 developers can use existing linkers, compilers, etc. to embed EXE format files in applications. Bon HEX-DUMP utility & HEX-ASCII formatter. No Royalties. $95. VISA, MC.
LUCTOR Corporation
21436 N. 20 Ave., Phoenix, AZ 85027
(602) 864-1298

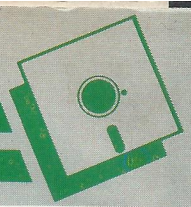
CIRCLE 98 ON READER SERVICE CARD

INSTANT ACCESS
to any directory without typing full path names. The COMMAND System automatically generates Short Names for every directory. Fast programs; many functions DOS forgot. Includes super programs to show files and directories. Fantabulous SEARCH Program that searches ASCII or Word Processor files. 30-day Money-Back Guarantee. Buy Now at the Special Introductory price of $75. Send Check or Call Now with your VISA or MC order. Free U.S. shipping.
CompuMagic Inc.
P.O. Box 437, Severn, MD 21144
(301) 969-8068

CIRCLE 99 ON READER SERVICE CARD

SCREEN MANAGER
MENU, WINDOW, and DATA ENTRY Support for the Professional Programmer! Interfaces to most languages. BASIC, C, FORTRAN, COBOL, Pascal, assembler. 100-Page Manual. 30-day money-back guarantee. No royalties. From $79. VISA/MC. CALL FOR FREE DEMO
The West Chester Group
P.O. Box 1304, West Chester, PA 19380
(215) 644-4206

CIRCLE 100 ON READER SERVICE CARD



FASTER THAN THE FASTEST C

- On the PC4000, run C four times faster than Microsoft C
- Full K&R standard C with extensions
- PC4000 PX/XT/AT board runs 5-10 MIPS
- PC4000 is 25 times faster than AT
- Ideal for image processing and control
- Forth and assembler also available
- From $1295 for PC4000, $595 for SC-C

Silicon Composers
210 California Ave., Ste.K
Palo Alto, CA 94306
(415) 322-8763

CIRCLE 101 ON READER SERVICE CARD

LEARN C ON TV

Learn C quickly and easily with the video training course "A Programmer's Introduction to C." Course includes 3½ hours of video instruction and a 115-page booklet. Cost: $400. A free 15-day review available.

Berkeley Decision/Systems
803 Pine St., Santa Cruz, CA 95062
(408) 458-0500

CIRCLE 102 ON READER SERVICE CARD

TURBO-TO-C TRANSLATOR

Turbo-to-C Tools $495:

- Pascal-to-C module maker
- Denesting Preprocessor
- 99+% Translator
- 4 C Source Libraries
- Tutorial manual
- 512K, MS-DOS compatible
- Other Pascals, too!

TGL Inc.
27096 Forest Springs, Corvallis, OR 97330
(503) 745-7476

CIRCLE 105 ON READER SERVICE CARD

CROSS ASSEMBLERS
UNIVERSAL LINKER, LIBRARIAN, PC/MS-DOS, VAX VMS/UNIX

- Fast Version 2.1
- Targets 30 microprocessors
- Macro, relocatable or absolute
- Binary or ASCII hexes
- Conditional assembly

Enertec Inc.
19 Jenkins Ave., Lansdale, PA
(215) 362-0966

CIRCLE 104 ON READER SERVICE CARD

TURBO PASCAL $2.50—DISK

TSS, the "BBS-by-mail." 135+ disks of compressed files. Membership fee ($25) includes starter package + 2 FREE disks with first order. Cat list $5. Anniversary sale: $250 for full membership PLUS full library through disk 150. VISA/MC/COD (non-U.S. add $10, U.S. funds only). Call our on-line BBS: (617) 545-9131. All files on line!

Turbo Source Search
P.O. Box 876, Scituate, MA 02066
(617) 545-6677

CIRCLE 103 ON READER SERVICE CARD

COMPILED BASIC TOOLS

We now carry a complete line of libraries and tools for compiled BASIC in addition to our FINALLY! series. Get our catalog of top brand products for compiled BASIC by calling:

(800) 423-3400 (9 a.m.-8 p.m. EST).
KOMPUTERWERK INC.
851 Parkview Blvd., Pittsburgh, PA 15215
PA & AK call (412) 782-0384

CIRCLE 106 ON READER SERVICE CARD

Write for
Computer Language

February: C
Deadline: Nov. 1

April: Debugging
Deadline: Dec. 15

May: Object-Oriented Programming
Deadline: Jan. 15

Send your proposal, outline, or draft to:
Regina Starr Ridley, Editor
Computer Language, 500 Howard St., San Francisco, CA 94105

The Window BOSS

Pop-up windows, pull down menus, status lines, and in context online help can be easily implemented. Your applications can drag windows around the screen and automatically sense the installed video card -- all this without snow, flicker or delay!

Tim Parker's column in the Feb. 1987 issue of Computer Language stated "the ease of use will cause some programmers to go overboard with windows", and "if you are a die-hard C programmer looking for a useful, license-free product, this is it".

COMPILERS SUPPORTED

Borland, Microsoft, Lattice, Computer Innovations, Datalight, Mark Williams Lets C

The Window BOSS supports PC/MSDOS for IBM PC/XT/AT and Compatibles

The Window BOSS will accelerate the development of system and application programs in the "C" Language. The BOSS will let you create programs that have the same look and feel as the top selling spreadsheet, database, and windowing products. All this and source code too for only $50.00 (Shareware disk $15.00).

Star Guidance Consulting, Inc.
273 Windy Drive, Waterbury, CT 06705
(203) 574-2449

CIRCLE 80 ON READER SERVICE CARD

NEVER GO BACK TO YOUR C REFERENCE MANUAL AGAIN!

AVDOC-C™ on-screen reference manual pops up on your PC screen whenever you need it! At the push of a button, you will have the information you need on C syntax, standard C library functions, and more. An on-screen calculator evaluates expressions for you in both decimal and hexidecimal and does ASCII character conversions. AVDOC-C gives you instant information on K & R standard C and ANSI extensions. Also available for 8051 and Z80 assembler languages.

ONLY $49.95

CALL 800-448-8500

AVOCET
SYSTEMS, INC.®
Software Development Tools for Industry

CIRCLE 81 ON READER SERVICE CARD

***NEW!* TLIB™ 4.0**
SOURCE CODE CONTROL
The best keeps getting better!

The critics *loved* TLIB 3.0...

"...packed with features... [generates deltas] amazingly fast... [of the 6 reviewed] the two best packages are Burton Systems' TLIB and [a $395 product], so designated because of their ease of use, abundance of features, and ability to be configured..." ***PC Tech Journal Sept 87***

"...has my highest recommendation." *Ronny Richardson,* ***Computer Shopper Aug 87***

- The **fastest, most powerful** system is now even faster!
- ***Many new features!*** Expanded keyword support. Multi-line comments. Branching, for multiple development lines. Extended wildcard and list-of-file support; creates lists by scanning source code for includes. Can merge (reconcile) multiple simultaneous changes and undo intermediate revisions. Network and WORM optical disk support.
- Includes a copy of Landon Dyer's excellent public domain **MAKE** utility (with source code for DOS & VAX/VMS).

PC/MS-DOS 2.x & 3.x ***Just $99.95*** *+ $3 s/h* Visa/MC

BURTON SYSTEMS SOFTWARE
P. O. Box 4156, Cary, NC 27519-4156
(919) 469-3068

CIRCLE 78 ON READER SERVICE CARD

BASIC
↓
C or Pascal

Ready to run, Syntax error free code, Re-Structured, Indented. Random file, Grapics, Free Library(Source code)
200 Page manual, Demo disk
BAS_C for Microsoft, Turbo, Lattice, etc
BAS_PAS for Turbo Pascal
GOTOLESS CONVERSION
Box 835910, Richardson, TX75083
Phone (214) 404 - 1404

CIRCLE 77 ON READER SERVICE CARD

PORTABLE WINDOWING SYSTEM

- **Develop windowing applications to run on a variety of operating systems.**
- **Currently running on UNIX System-V, Xenix, Unix BSD and MS-DOS operating systems.**
- **Compatible with System-V curses PLUS overlapping frames, resizing and moving frames/panels and much more. Includes a second program to describe various ASCII terminals.**
- **Binary $199. Source (in C) $299.**
- **Includes typeset manual with tutorial. No licensing fees. Visa, Amex, MC accepted.**

Call (416) 842 - 6873 For Free Demo.
PARKRIDGE COMPUTER SYSTEMS INC.

Inquire about our special MS-DOS Offer

Prices quoted in US Funds.

CIRCLE 82 ON READER SERVICE CARD

ADVERTISER INDEX

	PAGE NO.	CIRCLE NO.
Aker Corp.	27	21
Aldebaran Laboratories	29	22
Aldia Systems Inc.	118	92
ASCII	118	93
Aspen Scientific	21	18
Aspen Technologies Inc.	18	15
The Austin Code Works	72	33
The Austin Code Works	112	70
Avocet Systems	119	81
Berkeley Decisions/Systems	119	102
Blaise Computing	5	3
Blaise Computing	19	16
Block Island Technologies		41
Borland International	Cover II	—
Borland International	1	1
The Boston Software Works Inc.	94	51
Burton Systems Software	118	78
C Ware/DeSmet	110	—
CompuMagic	118	99
Conversational Computer System	118	88
Courseware Applications	57	27
Creative Programming	20	—
Database Programming & Design	104	86
Datalight	8	5
Ecosoft	81	31
Enertec Inc.	119	104
Essential Software Inc.	84	43
FairCom	87	46
Generics Software Ltd.	77	37
Gimpel Software	99	—
Glenco Engineering	9	6
Gotoless Conversion	118	77
Greenleaf Software Inc.	78	38
Gryphon Microproducts	112	72
Guidelines Software	17	14
Harvard Softworks	98	56
Hauppauge Computer Works Inc.	14	12
Haventree Software Ltd.	116	75
IMSI	Cover III	83
IMSL	67	40
Jou Laboratories	118	94
Jou Laboratories	118	95
Komputerwerk Inc.	119	106
Lattice	45	24
Lear Com Co.	118	90
Logitech Inc.	75	35
Lugaru Software Ltd.	97	54
Machine Independent Software Corp.	86	45
Mansfield Software	97	55
Manx Software Systems	13	11
Mark Williams Co.	Cover IV	84
Metagraphics	89	48
MetaWare Inc.	110	65
Microcompatibles	108	63
Microcomputer Tools Co.	118	97
Microsoft	33	—
Microsoft	74	—
Microsoft	105	—
Microsoft	107	—
Microsoft	109	—
Microsoft	111	68
Microsoft	113	—
MicroWay	95	52
N-TECH	112	69
Nanosoft	76	36
Next Generation Systems	114	73
Night Owl Computer Services	118	87
Norcom	103	59
Northwest Instrument Systems Inc.	16	13
Oakland Group	91	49
Oasys	115	74
Opt-Tech Data Processing	118	97
Parkridge Computer Systems Inc.	119	82
Polytron	10	7
Programmer's Connection	59-61	29
Programmer's Paradise	70,71	32
Programmer's Shop	92	50
Programmer's Shop	22,23	19
Q-N-E International	58	28
QCAD Systems Inc.	100	66
Quilt Computing	112	71
Radian	31	23
Raima Corp.	24	—
Rational Systems	110	57
Realia	102	58
The Report Store	118	86
Roundhill Computer Systems Ltd.	62	30
Santa Rita Software	6	4
SAS Institute	101	—
Scantel Systems Ltd.	120	85
Seidl Computer Engineering	96	53
Silicon Composers	119	101
Soft-Tel	118	89
SoftCraft Inc.	3	2
Softfocus	108	62
Software Developers '88	88	47
The Software Factory Inc.	12	9
Software Garden	85	44
Softway Inc.	108	64
Solid Software	110	67
Solution Systems	83	42
Solution Systems	106	61
Star Guidance Consulting	119	80
Sub-Systems	67	39
Synergy Systems	12	10
Tangent Designs	118	91
Tenon Software	67	—
Texas Instruments	69	—
TGL Inc.	119	105
True Basic	11	8
Turbo Power Software	117	76
Turbo Source Search	119	103
Visual Horizons	118	107
The West Chester Group	118	100
Whitesmiths Ltd.	46	25
Whitewater Group	54	26

CANADA'S SOURCE FOR C

– Canadian Sales
– Canadian Service
– Canadian Technical Support
– Canadian Product Knowledge

We specialize in programming & development software

LIFEBOAT • LATTICE • GREENLEAF • PHOENIX
SOFTCRAFT • MICROSOFT • BLAISE • ESSENTIAL
AGE OF REASON • DESMET • AZTEC
MARK WILLIAMS • GIMPEL • ROUNDHILL • GSS
HALO • FAIRCOM • RAIMA • INTEL • etc. • etc. •

VISA Call for full price list—Dealer enquiries welcome MasterCard

We know our products — we use them!

SCANTEL SYSTEMS LTD.

801 York Mills Rd., Don Mills, Ont., M3B 1X7
(416) 449-9252

The index on this page is provided as a service to our readers. The publisher does not assume any liability for errors or omissions.

TURBO PASCAL AND TURBO C... MEET

TURBOHALO

"Ideal! TurboHALO does the job comparable to packages costing $3000 to $4000."
Jim Bromley
Superintendent of Spectrum Management

"I like the speed of TurboHALO...it's ten times faster than the competition."
Deniz Terry
Doctoral Candidate

"TurboHALO is so fun... I use it to design programs as a hobby...It's got lots of ability."
William Porter
Control Systems Manager

"We evaluated all of the graphic development packages for Turbo Pascal, and TurboHALO was the hands down winner!"
Quinn Curtis
Largest New England Distributor

It's time to put graphics into your programming.
A picture's worth a thousand words. So your programming isn't complete until you have graphics. TurboHALO brings your screen and printer to life with subroutines that draw, chart, map, and display. All with color, shape, clarity, perspective and motion. With TurboHALO, create any picture you can imagine.

TurboHALO gives you graphics power.
TurboHALO gives you everything you need for Turbo C and Turbo Pascal graphics programming. A library of over 150 graphics subroutines. Drivers for over 42 graphics hardware devices for the IBM PC family and compatibles. You can create the images you want, on the hardware you have!

Fast, proven and reliable.
TurboHALO is up to ten times faster than other graphics toolkits. And with TurboHALO you get proven, reliable programming tools used by professionals for years.

You'll like TurboHALO or your money back!
TurboHALO is available for only $95. You get an unconditional 45-day money-back guarantee on TurboHALO. To order, call your dealer or IMSI at (415) 454-7101 or (800) 222-4723; in CA (800) 562-4723; in Washington DC (202) 363-9340 or in NC (919) 854-4674.

YES, *I want to see the difference TurboHALO makes in my graphics programming! Rush me the following TurboHALO Graphics Toolkit(s) @ $95 each plus $3 shipping. California residents add 6% sales tax.*

☐ *TurboHALO for Turbo Pascal*
☐ *TurboHALO for Turbo C*
☐ *Check enclosed for $ _____ (made payable to IMSI)*
☐ *Charge my credit card for $ _____* ☐ *VISA* ☐ *MasterCard*

Signature _____

Card Number _____ *Expiration Date* _____

Name _____

Title _____

Company _____

Street _____

City _____ *State* _____ *Zip* _____

IMSI™

CL1187

Mail to:
IMSI, 1299 Fourth Street, San Rafael, CA 94901

FOR GRAPHICS PROGRAMMING

TurboHALO requires 256K memory (min); memory resident drivers require 2K (Turbo Pascal only); DOS version 2.0 or higher; Borland language/compiler required. TurboHALO is a trademark of Media Cybernetics and IMSI. Turbo C and Turbo Pascal are trademarks of Borland.

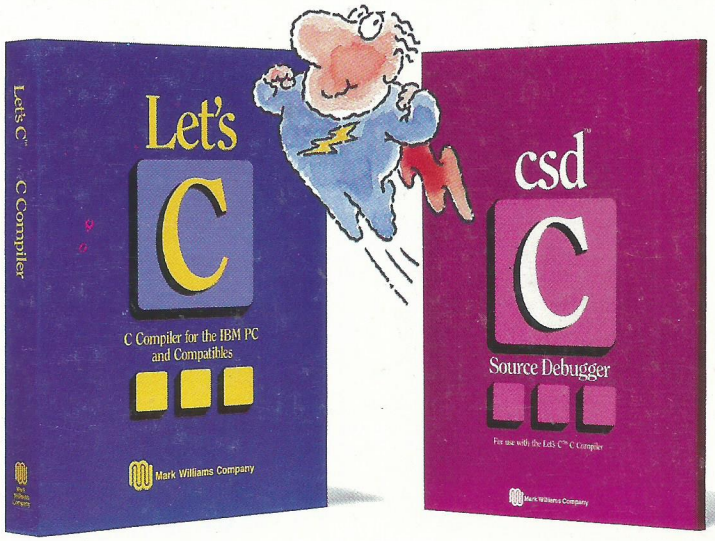
CIRCLE 83 ON READER SERVICE CARD

PRESENTING THE DIFFERENCE BETWEEN FAST COMPILING AND FAST PROGRAMMING.

For compiling speed, you can't do better than Let's C. But to really speed up programming you can't do without the powerful source level debugger, *csd*.

If you want the power, portability and flexibility of C, start with the complete compiler, Let's C. For utilities, editor, compiling speed and fast, dense code, Let's C has it all.

But to get your programs up and running you need more. Because even the fastest compiler can't outrun bugs. You need the revolutionary C Source Debugger, *csd*.



CUT DEVELOPMENT TIME IN HALF WITH *csd*

csd lets you bypass the time consuming frustrations of debugging–like long dumps and clunky assembler. With *csd*, you actually debug in C. You learn faster because you watch your program run in C. You finish faster because *csd* combines the speed of a compiler with the interactive advantages of an interpreter. The end result? Development time is sliced in half.

LIMITED TIME OFFER FREE *csd* WITH LET'S C!

REVIEWERS ARE RAVING ABOUT LET'S C AND *csd*.

"Let's C is an inexpensive, high-quality programming package... with all the tools you will need to create applications."
–William G. Wong, *BYTE*, August 1986.

"The performance and documentation of the $75 Let's C compiler rival those of C compilers for the PC currently being sold for $500...highly recommended..."
–Marty Franz, *PC TECH JOURNAL*, August 1986.

"csd *is close to the ideal debugging environment...a definite aid to learning C and an indispensable tool for program development."*
–William G. Wong, *BYTE*, August 1986.

"This is a powerful and sophisticated debugger built on a well-designed, 'serious' compiler."
–Jonathon Sachs, *Micro/Systems Journal*, April, 1986

LET'S C AND *csd* FEATURES

NEW VERSION 4.0!

Let's C:

- Now compiles twice as fast
- Integrated edit-compile cycle: editor automatically points to errors
- Includes both small and large memory model
- Integrated environment or command line interface
- 8087 sensing and support
- Documentation features new lexicon format
- MS-DOS object compatible
- New make utility
- Fast compact code plus register variables
- Full Kernighan & Ritchie C and extensions
- Full UNIX compatibility and complete libraries
- Many powerful utilities including make, assembler, archiver, cc one-step compiling, egrep, pr, tail, wc
- MicroEMACS full screen editor with source included
- Supported by dozens of third party libraries
- For the IBM-PC and Compatibles
- Not copy protected

Sieve Benchmark
(Compile time in seconds)
Let's C: **2.8** (On 512K 6Mhz IBM-AT)
Turbo C: **3.89** (As advertised)

csd:

- Large and small memory model
- Debug in C source code, not assembler
- Monitor variables while tracing program
- Does not change program speed or size
- Provides separate source, evaluation, program and history windows
- On-line help screens
- Can interactively evaluate any C expression
- Can execute any C function in your program
- Trace back function
- Ability to set trace points
- Not copy protected

START TO FINISH, THERE'S NO BETTER ENVIRONMENT.

Get started with the right C compiler and you'll have everything you need for development–including source level debugging. On top of it all, Let's C and *csd* are today's best values in professional C programming tools. And most reliable: Mark Williams C compilers have been sold with DEC, Intel and Wang computers since 1981.

60 DAY MONEY BACK GUARANTEE

Mark Williams gives you a full 60 days to find out just how good Let's C and *csd* really are–or your money back.

So if you want more than a fast compiler–if you want your programs up and running fast, ask for Let's C and *csd*. You'll find them at your software dealer's, in the software department of your favorite bookstore, through the Express Program at over 5500 Radio Shacks or you can order now by calling **1-800-MWC-1700.***

*In Illinois call, 1-312-472-6659.

Mark Williams Company
1430 West Wrightwood, Chicago, Illinois 60614

© 1987 Mark Williams Company
Let's C is a registered trademark of the Mark Williams Company.
UNIX is a trademark of Bell Labs.

MARK WILLIAMS LET'S C AND *csd*. ONLY $75 EACH.

CIRCLE 84 ON READER SERVICE CARD